Universidade Federal de Santa Catarina

Acesso remoto a horários de ônibus

Uma abordagem de acesso a dispositivos móveis utilizando J2ME e MIDP

Ramon Hugo de Souza

Florianópolis, Dezembro de 2004

Universidade Federal de Santa Catarina
Departamento de informática e estatística
Curso de Ciências da Computação

Acesso remoto a horários de ônibus
Uma abordagem de acesso a dispositivos móveis utilizando J2ME e MIDP

Ramon Hugo de Souza

MONOGRAFIA apresentada ao Curso de Bacharelado em Ciências da Computação do Departamento de Informática e Estatística da UNIVERSIDADE FEDERAL DE SANTA CATARINA, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação

Orientador: Prof. Mario Dantas, P.H.D. Southampton
Palavras-Chave: Dispositivos Móveis, Refatoração

Florianópolis, Dezembro de 2004

Acesso remoto a horários de ônibus

Uma abordagem de acesso a dispositivos móveis utilizando J2ME e MIDP

Por Ramon Hugo de Souza

Monografia defendida e aprova em 9 de novembro de 2004, pela banca examinadora constituída pelos seguinte integrantes:

Prof. PHD Mario Dantas

Prof. Dr. Vitório Bruno Mazzola

Prof. Dr. José Eduardo De Lucca

Um filósofo de estatura imponente não pensa em um vazio. Mesmo suas idéias mais abstratas são, em alguma medida, condicionadas pelo que é ou não é conhecido na época em que ele vive.

- Alfred North Whitehead

Agradecimentos aos amigos Rodrigo Gonçalves (UFSC-01/1) pela ajuda na resolução das dúvidas referentes à banco de dados, a Guilherme Artur Fucks Martins (UFSC-01/2) pela ajuda referente à tecnologia MIDP, a Erich Dimitry Lunardeli Silvestre (UFSC-04/1) pela ajuda com o programa para análise dos dados, a Alex Sandro Roschildt Pinto (UFSC) pela ajuda quanto à resolução do problema e ao professor Mario Dantas (PHD Southampton 1997) pelas idéias e orientação do trabalho.

Sumário

Lista de Abreviaturas

API	Application Program Interface
CDC	Connected Device Configuration
CDMA	Code Division Multiple Access
CLDC	Connected, Limited, Device Configuration
GNU	GNU's Not Unix
LGPL	Lesser General Public License
GPRS	General Packet Radio Service
GSM	Global System For Communication
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure HTTP
IP	Internet Protocol
J2ME	Java 2 Micro Edition
JVM	Java Virtual Machine
MIDP	Mobile Information Device Profile
PDA	Personal Digital Assistant
PDAP	Personal Digital Assistant Profile
PKI	Public Key Infrastructure
RMI	Remote Method Invocation
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TDMA	Time Division Multiple Access
UFSC	Universidade Federal de Santa Catarina
XP	Extreme Programming

Lista de Figuras

Resumo

A computação móvel, que já é parte do cotidiano, tende a cada vez mais integrar serviços que levem aos seus usuários uma gama de informações que lhes podem ser úteis quanto a resolver problemas característicos do seu dia a dia. Tecnologias que não favorecem uma integração de conexão ativa tendem a desaparecer, e os dispositivos tendem a se tornar todos integrados em uma rede de abrangência mundial como a Internet. Mesmo em se tratando de redes menores, a tendência é que os dispositivos tenham todos a possibilidade de comunicação através de redes privadas, podendo assim um acessar informações de outrem, e também possibilitando a utilização de processamento distribuído e abrindo as possibilidades para diversas aplicações na área de jogos de entretenimento.

O estudo de uma abordagem de acesso a dispositivos móveis tem como foco as tecnologias utilizadas na época do estudo relativas a dispositivos de pequeno porte, sendo a aplicação em específico desenvolvida para dispositivos celulares. O objetivo foi demonstrar como a integração de serviços necessários ao dia a dia poderia vir a ser muito melhor utilizada quando acessada por um dispositivo que o usuário leva sempre consigo. Foi criado um banco de dados experimental possuindo dados reais relativos ao transporte coletivo da cidade de Florianópolis – SC, e a partir deste foi desenvolvido um acesso via um servidor específico para tratar as requisições dos dispositivos – devido ao fato destes não possuírem todas as possibilidades de tratamento de informações como quando se tratando de uma abordagem em ambientes desktop.

Inicialmente foram levantados os dados reais os quais seriam a base para demonstração de um aplicativo para o devido estudo. Para uma clara noção dos requisitos necessários à aplicação foi criado um software para desktop, o qual permitiu uma melhor modelagem de como o sistema viria a se desenvolver. O simulador provido pelo toolkit da própria Sun foi utilizado para simular o ambiente de dispositivos celulares. As pesquisas requisitadas pelo dispositivo foram tratadas através de conexões por sockets com o servidor que é quem na verdade faz a pesquisa e em tempo de execução gera objetos persistentes em relação a esta pesquisa. Porém uma forma alternativa de envio

de informações foi desenvolvida devido a limitações da tecnologia atual para desenvolvimento de software para pequenos dispositivos.
Aplicações de processamento distribuído e frameworks para acesso a informações em banco de dados de maneira que se possa driblar os problemas inerentes a limitação dos dispositivos são maneiras interessantes de se desenvolver novos softwares relativos tecnologias móveis. O software gerado ao fim do estudo se mostrou de grande utilidade, e poderia realmente ser aplicado à rede de transportes da cidade onde foram obtidos os dados – ou qualquer outra cidade, já que o banco de dados foi desenvolvido de maneira a possibilitar até mesmo algumas pesquisas mais apuradas que não foram implementadas. Um banco de dados distribuído por região poderia ser criado, e até mesmo linhas entre cidade poderiam ser adicionadas. A pesquisa poderia ser utilizada de forma eficiente no nível de pesquisa de banco de dados até mesmo quando se considerando todos os caminhos pelo país.

Palavras-Chave: Dispositivos Móveis, Refatoração, Computação Móvel, Sistemas Distribuidos.

Abstract

The mobile computation, that is already a part of our daily, tends to integrate services that take to its users a lot of information that can be useful to decide characteristic problems of its day by the day. Technologies that do not favor an integration of active connection tend to disappear, and the devices tend to become all integrated in a net of world-wide size as the InterNet. When considering lesser nets, the trend is that the devices have all the possibility of communication through private nets, thus being able one to have access information of the others, and also making possible the use of distributed processing and opening the possibilities for diverse applications in the area of entertainment games.

The study of an access boarding to mobile devices have as focus the existing technologies at the time of the study of the devices of small size, being the developed application in specific for cellular devices. The objective was to demonstrate how the integration of necessary services to the day by the day could be better used when the user had access for a device that is always with him. An experimental data base was created with real data from the collective transport of the city of Florianópolis - SC, and based on it was developed a specific server application to deal with the solicitations of the devices – because the fact of these devices not to possess all the possibilities of treatment of information as when talking about desktop environments.

Initially the real data had been raised which would be the base for demonstration of an application for the study. For a clear notion of the necessary requirements to the application a software for desktop was created, which allowed one better modeling of as the system would come to be developed. The simulator provided for toolkit of the Sun was used to simulate the environment of cellular devices. The researches requested for the device had been treated through connections by sockets with the server who in the fact who makes the research and in execution time generates persistent objects to this research. However an alternative form of sending information was developed because of limitations of the current technology for development of software for small devices.

Applications of distributed processing and frameworks for access the information in data base thus can dribble the inherent problems of the limitation of the devices are new interesting ways of developing softwares relative to mobile technologies. The software generated in the end of the study were showed great utility, and could really be applied to the net of transports of the city where the data came from - or any other city, since the data base was developed in way to make possible some more refined researches even though they had not been implemented. A data base distributed by region could be util, and even though lines between cities could be added. The research could even though be used of efficient form in the level of research of data base when considering all the ways around the country.

Key-Words: Mobile Devices, Refactoring, Mobile Computation, Distributed Systems.

Introdução

Este trabalho teve como base principal da sua motivação a busca de integração de serviços cotidianos – que são providos ou mesmo ainda não providos por outros meios – à realidade dos dispositivos móveis.

Partindo-se do princípio de que um celular – por exemplo – utilizando a tecnologia bluetooth[1] poderia acessar uma rede wireless local, como uma rede residencial, ou mesmo dados em outro dispositivo celular para troca de informações, e então a partir desta funcionar até mesmo como um dispositivo cem por cento conectado a Internet, resolveu-se considerar a utilização de tecnologias já existentes, e já utilizadas em conjunto com estes dispositivos, como a rede de serviços provida pelas empresas de telefonia celular, para que fosse possível se integrar serviços on-line para busca de informações com estes dispositivos.

De uma forma geral a motivação para este trabalho foi a busca de facilidades, como obtenção de horários de ônibus, relacionadas ao dia a dia integradas na computação móvel.

A aplicação desenvolvida no estudo teve como objetivo também demonstrar que muitos outros serviços podem ser desenvolvidos em camadas intermediárias da comunicação entre estes dispositivos e o seu servidor de dados – por exemplo.

Existem inúmeros trabalhos que podem ser realizados em relação a esta nova tecnologia, muitas áreas de abordagem como segurança e confiabilidade ainda precisam ser melhor exploradas em relação à estes dispositivos, e pretende-se demonstrar não só as possibilidades, mas também o que seriam hoje necessidades quanto a clientes que utilizam serviços providos por dispositivos móveis.

Esta é uma tecnologia que vem em pouco tempo ganhando importância no meio de desenvolvimento de sistemas e que ainda possui uma gama de portas abertas a trabalhos científicos a serem realizados quanto a sua utilização e mesmo melhoramento do modo como é feita a comunicação – em relação à segurança e a confiabilidade dos serviços, por exemplo.

[1] Bluetooh é uma tecnologia de comunicação por ondas de radio de pequena distância que visa simplificar as comunicações entre a Internet e dispositivos que implementam a tecnologia.

Este estudo inicia com uma abordagem geral sobre dispositivos móveis no primeiro capítulo, de modo a demonstrar as tecnologias vigentes do mercado – as quais fornecem o serviço de conexão ativa como considerado inicialmente, partindo-se então para uma abordagem sobre as tecnologias utilizadas para o desenvolvimento da aplicação.

O segundo capítulo se refere à parte de modelagem do sistema em si, e de como se buscou uma forma científica de se desenvolver o software que se enquadrasse na necessidade de um desenvolvimento com visualizações rápidas de partes do sistema já em funcionamento, abordando programação extrema e refatoração de código.

No terceiro capítulo é apresentado o modo como foi desenvolvido o protótipo utilizado para se levantar os requisitos iniciais do sistema e como ele levou a especificação inicial do sistema, já demonstrando uma aplicação funcional com um banco de dados sendo desenvolvido em paralelo com o propósito de se demonstrar uma aplicação com resultados já consistentes para então se chegar na especificação final e o desenvolvimento do sistema final.

O quarto capítulo se refere às conclusões obtidas com relação ao trabalho e faz menção a possíveis trabalhos futuros na área e em relação ao software desenvolvido.

Capítulo 1 – Dispositivos Móveis

Dispositivos móveis vêm ganhando cada vez mais importância no nosso dia a dia, e não apenas para a simples comunicação por voz, mas também cada vez mais e mais para serviços que são prestados para estes dispositivos.

A praticidade de se ter à mão um dispositivo que possa prestar vários serviços úteis ao nosso cotidiano tem levado à rápida ascensão das tecnologias desenvolvidas para tais dispositivos.

Como dispositivos móveis considerados neste estudo levou-se em conta celulares e PDAs – dispositivos de pequeno porte pessoais, sendo que as tecnologias relacionadas à PDAs apresentadas são apenas de modo ilustrativo já que as tecnologias que desejasse abordar se referem a uma abordagem voltada especificamente a dispositivos celulares.

1.1 Tecnologias Wireless

Algumas tecnologias referentes a dispositivos móveis são usadas hoje nos dispositivos celulares do mercado, porém algumas delas tem uma grande tendência a desaparecer nestes dispositivos – como no caso dos dispositivos de conexão que não seja cem por cento ativa – por desfavorecerem o desenvolvimento de software que possa levar a uma evolução dos dispositivos.

Informações sobre as tecnologias segundo [CARNIEL 2003].

1.1.1 TDMA – Time Division Multiple Access

O acesso múltiplo por divisão de tempo é uma tecnologia que não diferencia voz de dados. Ou seja, no caso de acesso à Internet com dispositivos que utilizem esta tecnologia o preço pago seria deveras desconfortável para o cliente do serviço. Além do que a conexão não ultrapassa os 9600bps.

1.1.2 CDMA – Code Division Multiple Access

O acesso múltiplo por divisão de códigos é uma tecnologia que diferencia voz de dados. É uma tecnologia de conexão cem por cento ativa, ou seja, não existe delay de conexão. Além de que a taxa de transmissão pela conexão é de 256Kbps segundo a CDMA 1xRTT.

1.1.3 GSM – Gobal System For Communication

O sistema global de comunicação é também uma tecnologia que diferencia voz de dados, sendo como o CDMA também cem por cento conexão ativa. A única diferença desta tecnologia para a CDMA é que esta opera por GPRS - General Packet Radio Service é o padrão para conexões wireless que funcionam a velocidades maiores que 115Kb/s.

1.2 Java e a API J2ME

A API Java 2 Micro Edition é uma API voltada para o desenvolvimento de aplicativos para dispositivos de pequeno porte, como no caso de dispositivos de comunicação sem fio como celulares e PDAs.

O J2ME pode ser subdividido em configurations, profiles e APIs (Application Program Interface) – definição da interface de programação da aplicação – adicionais.

Os profiles são definições mais específicas que as configurations, sendo estes formas mais concretas das mesmas. Cada profile definido em J2ME é um tipo de configuration – como será definido a seguir.

As APIs são um tipo específico de profile em especial. Algumas API's adicionais – além das fornecidas pela API J2ME - podem ser fornecidas pelos fabricantes dos dispositivos para uma melhor utilização dos mesmos – porém softwares desenvolvidos segundo este paradigma perdem a característica de portabilidade entre dispositivos.

Informações de tecnologia J2ME segundo [CARNIEL 2003].

1.2.1 Configurations

Nesta API Java temos duas configurations definidas para aparelhos de diferentes portes. A primeira é o CLDC – Connected, Limited, Device Configuration – que rege as configurações para os dispositivos realmente pequenos, como celulares e PDAs. A segunda é o CDC – Connected Device Configuration – que define as configurações para aparelhos não tão pequenos quanto os definidos pelo CLDC, porém ainda de pequeno porte e de baixa capacidade de processamento.

1.2.1.1 CLDC – Connected, Limited, Device Configuration

O CLDC foi desenvolvido para dispositivos com memória válida para Java de 160KB até 512 KB. Ou seja, dispositivos extremamente pequenos. Sendo que o processamento destes dispositivos dificilmente ultrapassa a freqüência de 10 Mhz, e a velocidade de conexão também pode não ultrapassar os 9600bps – como no caso de utilização com tecnologia TDMA.

1.2.2 Profiles

Podem haver vários profiles definidos nas especificações J2ME, porém nosso foco se dará nos profiles que ficam acima do CLDC.

Acima do CLDC dois profiles se destacam em importância, sendo eles o MIDP – Mobile Information Device Profile – que é uma especificação para dispositivos móveis como celulares, e o PDAP – Personal Digital Assistant Profile – que é uma especificação desta configuration para PDAs.

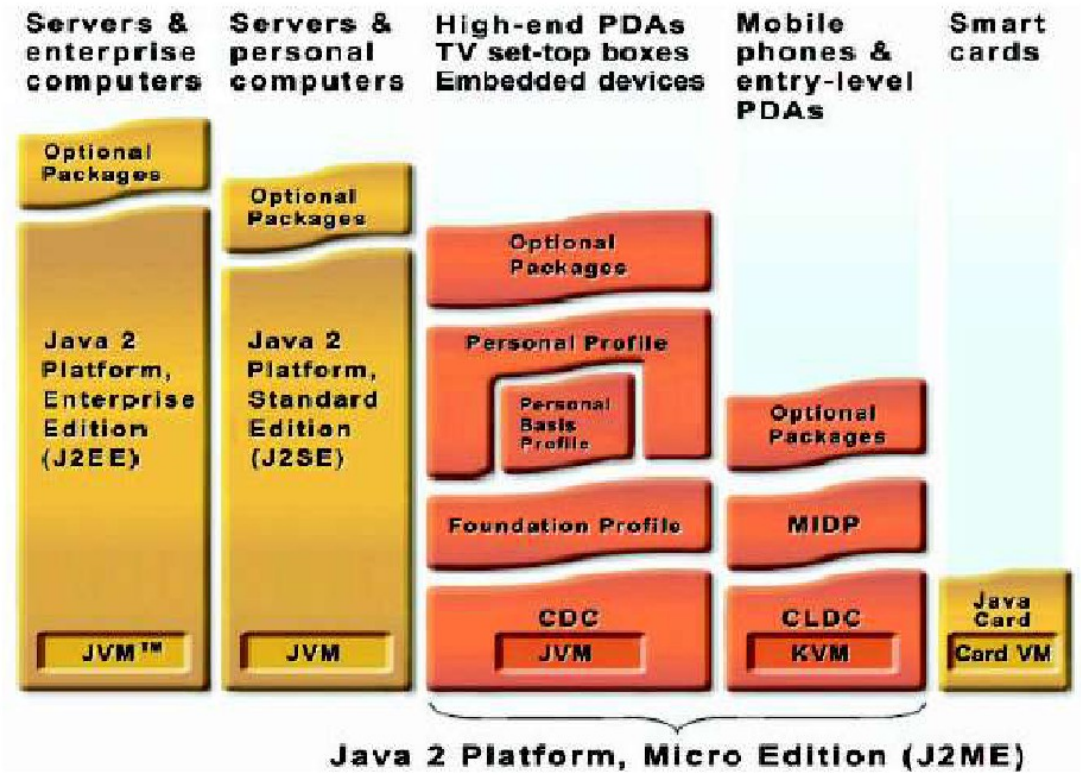


Figura 1 - Esquema de tecnologias baseadas em diferentes máquinas virtuais Java (De [CARNIEL 2003]).

1.2.2.1 MIDP – Mobile Information Device Profile

As especificações do MIDP – o principal profile para desenvolvimento de aplicações para dispositivos celulares – são as seguintes:

- 128KB de memória não volátil Java;
- 32KB de memória volátil para tempo de execução;
- 8KB de memória não volátil para armazenamento de dados;
- Tela de tamanho mínimo de 96x54 pixels;
- Capacidade de entrada de dados pelo teclado do celular, ou mesmo um teclado externo, ou touch screem;
- Possibilidade de envio e recepção de dados através de conexão.

Um adendo importante é que o Java não tem acesso a API específica do celular em si. O que o Java faz é acessar algumas funções específicas do aparelho – isso por motivos óbvios em relação à segurança.

Os dados que são restritos do acesso Java podem ser disponibilizados, porém, no caso de o fabricante do aparelho disponibilizar a API específica. Antes da versão 2.0 do MIDP esta era uma prática comum quando do desenvolvimento de jogos.

A maquina virtual tem espaço independente de memória, e não pode acessar a memória das aplicações nativas do celular.

1.2.3 API’s

Como explanado anteriormente, as APIs são implementações específicas dos profiles. Sendo estas – no caso da tecnologia Java – as classes que se pode usar para desenvolver software para os dispositivos que possuem suporte a tecnologia especificada.

1.2.3.1 APIs do MIDP 1.0

Na primeira versão do MIDP não existiam APIs específicas para o desenvolvimento de jogos, conexões seguras e o tratamento de operações multimídia também deixava a desejar.

Estas são as APIs implementadas na primeira versão do MIDP:

· java.lang
· java.lang.ref (somente 1.1)
· java.io
· java.util
· javax.microedition.io
· javax.microedition.lcdui
· javax.microedition.midlet
· javax.microedition.rms

Sendo que para algumas aplicações era necessário que se obtive as APIs disponibilizadas pelos fabricantes dos dispositivos.

A versão 1.0 do MIDP não possui suporte a ponto flutuante, o que foi corrigido com a inclusão da API java.lang.ref quando da versão 1.1 do CLDC. Na versão 1.0 eram utilizados alguns programas auxiliares para o uso de ponto flutuante.

Algumas outras restrições da versão 1.0 segundo [CARNIEL 2003] são:

– Sem user classLoading;
– Sem finalização de objetos (finallize() de java.lang.Object);
– Garbage collector existe, porém não executa método automático;
– Sem RMI[2] e sem JINI[3];
– Sem métodos nativos, a não ser os fornecidos pela JVM nativa;
– MultiThreading (sem interrupt(), pause(), resume() e stop());

[2] Remote Method Invocation – conjunto de protocolos desenvolvido pela divisão Javasoft da SUN que permite que objetos Java se comuniquem remotamente com outros objeto Java.
[3] JINI – Software da Sun Microsystems que busca simplificar a coneção e compartilhamento de dispositivos como impressoras e drives de disco em rede.

– Sem thread groups e thread Deamons.

1.2.3.2 APIs do MIDP 2.0

A segunda versão do MIDP já implementa conexão segura utilizando SSL (Secure Sockets Layer) – conexão por uma camada de socket segura – por HTTPS (Secure http) – conexão http segura usando SSL, bem como APIs específicas para o desenvolvimento de jogos sem que seja necessário se conhecer a API específica do aparelho – o que acabava com a portabilidade do software.

Estas são as APIs implementadas na segunda versão do MIDP:

· java.lang
· java.lang.ref (a partir da versão 1.1)
· java.io
· java.util
· javax.microedition.io
· javax.microedition.lcdui
· javax.microedition.lcdui.game (a partir da versão 2.0)
· javax.microedition.media (a partir da versão 2.0)
· javax.microedition.media.control (a partir da versão 2.0)
· javax.microedition.midlet
· javax.microedition.rms
· javax.microedition.pki (a partir da versão 2.0)

O MIDP 2.0 se aproxima mais do J2SE, com Permission Types, Protection Domains baseados em IP's[4] e PKI's[5].

[4] Internet Protocol – Protocolo Internet utilizado na camada de Rede segundo o modelo OSI ou na camada Internet segundo o modelo Internet.
[5] Public Key Infrastructure – Estrutura de Chave Pública, um sistema de certificados digitais, autoridades certificadoras, e outras autoridades registradoras que verificam e autenticam a validade da transação na Internet.

Capitulo 2 – Modelagem

2.1 Extreme Programming

Extreme Programming é uma abordagem alternativa aos métodos de desenvolvimento clássicos de engenharia de software, com procedimentos de menor complexidade e custo do que as alternativas clássicas. Baseia-se na idéia básica de programação feita por pequenas equipes, que lançam versões incrementais de pequenos programas. Simplicidade nos métodos de programação, forte comunicação entre desenvolvedores e usuários finais e realimentação entre os desenvolvedores são encorajadas pelo método.

XP surgiu à cerca de oito anos como uma alternativa a atender a crescente necessidade do mercado de uma eficácia e eficiência maior no resultado de projetos de software.

Os processos de desenvolvimento no caso da utilização desta abordagem necessitam ser melhorados em relação aos processos de abordagens mais usuais de engenharia de software. Porém a única diferença vem a ser a forma de implementação destes.

Os parágrafos seguintes baseiam-se em [EXTREME 2004] caso não haja indicação contrária.

2.1.1 Do nascimento da engenharia de software ao XP

A idéia central da engenharia de software surgiu em 1968 quando Edsger Djikstra escreveu uma carta intitulada “GOTO Statement Considered Harmful”. Nesta época acreditava-se que metodologias maiores e mais bem disciplinadas seriam a

solução aos problemas dos softwares desenvolvidos. Até então era pratica comum os programadores desenvolverem códigos de qualquer maneira. O que tornava os programas difíceis de entender e normalmente cheios de erros.

Na década de oitenta então foram criadas regras e mais regras para o desenvolvimento de software, o que parecia estar levando cada vez mais ao ponto de se conseguir de forma parametrizada como se desenvolver softwares no tempo necessário sem que maiores problemas fossem encontrados – como ininteligibilidade do código do software.

Chegamos ao século vinte e um com um conjunto de regras difíceis de seguir, procedimentos de complexo entendimento e muita documentação escrita de forma abstrata. Criando software que nos ajudava a criar software acabamos por perder o controle sobre isso. Surgiram então outra vez os programadores que programavam de qualquer maneira.

Então estas complexas regras começaram a serem utilizadas de forma simplificada. E então com algumas regras e praticas leves e concisas surgiu o XP.

2.1.2 Práticas de planejamento

Descrições textuais do usuário são a base do planejamento em XP. O escopo do projeto é eficientemente criado com base no modo correto de utilização das descrições feitas pelo usuário. Estas serão aqui chamadas de “User Stories”.

2.1.2.1 User Stories

Servem ao mesmo propósito dos use cases, mas não são a mesma coisa.

Os usuários finais os escrevem descrevendo coisas que eles gostariam que o sistema fizesse. Eles servem para estimar o plano de entrega de versões.

As “users stories” servem também para que sejam criados testes de aceitação do software. Testes que indiquem se o software faz mesmo o que o cliente pediu no seu plano inicial do sistema.

Estas descrições do sistema devem possuir detalhes de forma a minimizar o risco da produção do software de modo a se poder estimar quanto tempo será necessário para que se implemente a descrição. Não é preciso que se restrinja a descrição apenas como a interface do usuário.

As definições dadas pelos usuários precisam estar definidas para que se possa preparar o plano de versões a serem demonstradas para o mesmo. O foco das user stories está nas necessidades do usuário final.

2.1.2.2 Small Releases

Pequenas versões do sistema devem ser desenvolvidas baseadas no planejamento de versões para que o usuário possa avaliar o sistema e se necessário ele possa ser mudado durante o seu desenvolvimento sem um impacto posterior maior sobre o sistema como um todo.

2.1.2.3 Desenvolvimento iterativo

O desenvolvimento iterativo serve para dar agilidade ao processo de desenvolvimento. Os prazos de entrega de versões devem ser constantes para que o desenvolvimento da aplicação segundo esta abordagem possa ter uma melhor funcionalidade.

2.1.2.4 Planejamento Iterativo

Um planejamento iterativo é feito ao início de cada iteração para produzir um plano de iterações para as tarefas de programação. Testes de aceitação que precisam ser refeitos são selecionados.

Talvez a cada três ou quatro iterações possa ser necessário se rever o plano de tempo estimado para a produção da versão. Não se deve tentar acelerar o processo para que se obtenha o resultado no tempo estimado, e sim se deve analisar novamente o custo de cada componente envolvido.

2.1.2.5 Simplicidade

A abordagem XP parte do ponto que uma modelagem simples sempre ira tomar menos tempo que uma modelagem complexa. Sendo sempre mais rápido e barato repor um código com problema agora do que deixar para resolver o problema futuramente.

2.1.3 A arquitetura do sistema

Os desenvolvedores se utilizam de protótipos para obter um design simples do que viria a ser o sistema. A abordagem XP encoraja todos os membros a participar do desenvolvimento do modelo e a entenderem o mesmo.

2.1.4 Métodos de criação de código

Devido à suas características como um modo geral a qualidade de código no modelo XP é um fator de grande importância.

XP encoraja a programação em pares, utiliza-se da técnica de refatoração de código e tem como importante tarefa a criação de testes para o sistema.

A parte de testes possui grande importância no modelo, já que esta abordagem se propõe a entregar o software ao usuário final sem erros.

2.2 Refactoring

Os parágrafos seguintes baseiam-se nos trabalhos de [FOWLER 2003], [CINNÉIDE 2003], [TOCUDA 2001] e [CASTOR 2003].

Refatoração é uma mudança feita na estrutura interna do software que o melhora de alguma maneira, mas não altera seu comportamento. O avanço que se espera com refatoração é algo como editar um diagrama da classe ser tão simples quanto adicionar uma linha entre classes para representar um relacionamento ou mover por herança uma variável de uma subclasse para superclasse. Sua importância aumentou como uma técnica para melhora de projeto de código existente, em especial com o advento do uso de metodologias como Extreme Programming que envolve pouco projeto e iterações múltiplas com o ciclo de vida do software.

2.2.1 Refatoração de estruturas de classes

Refatoração de estruturas de classes é uma forma parametrizada de transformação de programa – classes no caso específico de orientação a objetos – que preserva o seu comportamento e atualiza de forma automática o projeto de uma aplicação e seu código fonte. Esta transformação costuma ser simples, como um impacto direto, o que não necessariamente signifique que este seja trivial, sobre o código fonte. “A refatoração é mais precisamente definida por (a) um propósito, (b) argumentos, (c) uma descrição, (d) condições de acionamento, (e) um estado inicial e (f) um estado final” [TOKUDA, 2001]. A aplicação de refatoração tem a vantagem de trabalhar com a arquitetura do design de um código existente no nível de diagramas de classes deixando o nível do código a ser refatorado para a parte automatizada.

Para enfatizar o uso de refatoração vamos tomar como exemplo o modelo de projeto em cascata. Considerando o modelo proposto deve ser feita coleta de requisitos e então segue-se para e sua análise. Passa-se então a etapa de projeto do sistema, e em seguida vai-se para a etapa de codificação. Por fim são feitos os testes. Em teoria tudo

parece correr muito bem, porém problemas inerentes podem aparecer. O não entendimento do problema, ou melhor, o entendimento de forma errônea do problema proposto, ou o não entendimento dos requisitos definidos pelo usuário podem tornar o projeto original inadequado.

Neste caso podem ser necessárias modificações no projeto, mesmo com código já pronto, gerando mais código posteriormente. O código do programa pode então vir a se tornar confuso.

A refatoração pode ser utilizada para tornar o programa mais reutilizável e fácil de ser compreendido. O software pode ser restaurado por meio da aplicação de várias refatorações, sem que seu comportamento seja modificado.

A refatorção proporciona uma melhoria no design, uma maior facilidade no entendimento do software e torna-se mais simples a localização e correção de bugs.

No exemplo anteriormente mencionado pode ser necessário, por exemplo, ser adicionada uma nova funcionalidade ao software que não foi definida no projeto inicial. Ou pode ser necessária a correção de um bug, ou mesmo apenas a revisão do código. Em todos estes casos o uso de refatoração se torna interessante, afinal o projeto – ou mesmo o nível de reutilização do software – pode vir a ser prejudicado por tais modificações.

A utilização de refatoração se torna bastante visível quando da utilização de Extreme Programming, onde é mais fácil que uma funcionalidade passe sem que se perceba. O pouco projeto neste tipo de programação pode fazer com que alguns requisitos não sejam inicialmente notados, mas as múltiplas iterações com o ciclo de vida do programa cooperam para que as novas implementações sejam facilmente adicionadas dentro deste paradigma de programação. Assim pode-se chegar a um ponto que a refatoração se torna visivelmente necessária.

A refatoração impõe alguns obstáculos ao programador, como a dificuldade de entender um projeto de sistema, ou o perigo de introdução de novos bugs. Porém sua não utilização pode deixar o design confuso, e o código pode se manter frágil. Essas duas características colaboram claramente para que mudanças se tornem freqüentes e caras. Afinal o projeto pode não ser adequado, e quando da necessidade de modificações a falta de um design bem definido pode prejudicar em muito a correção dos erros e tornar caras essas modificações.

É claro que no caso de um projeto que não corresponda ao modelo necessário pode se tornar inviável a utilização de refatoração – quando do caso de necessidade de se refazer a modelagem completamente.

Retomando a automatização deve-se, idealmente, provar de modo formal que o comportamento é preservado quando do uso de refatoração. Na prática isto geralmente não é feito. Primeiro porque automatizar refatorações requer um esforço de engenharia significativo quanto à necessidade de se construir compiladores que permitam que se manipule árvores de sintaxe abstratas dos programas e que dêem facilidades de metaprogramação convenientes para a geração do código. E esses compiladores quando feitos em C++, por exemplo, se tornam meio assustadores. Em segundo, provar que as refatorações preservam o comportamento requer uma semântica formal para a linguagem a ser definida. Este é um esforço demasiadamente grande, e muito além do que se deseja realizar.

2.2.2 Aplicações

Alguns cuidados devem ser tomados quando da utilização de refatoração. Ela deve ser feita independente de quaisquer outros processos. Deve ser feita passo a passo, sempre recompilando-se o código e testando. A cada passo o comportamento do sistema deve ser mantido. Não se pode esquecer que as modificações afetam o nível de projeto e de codificação do programa.

Na figura 2 podemos ver um exemplo simples de refatoração por adição de superclasses.

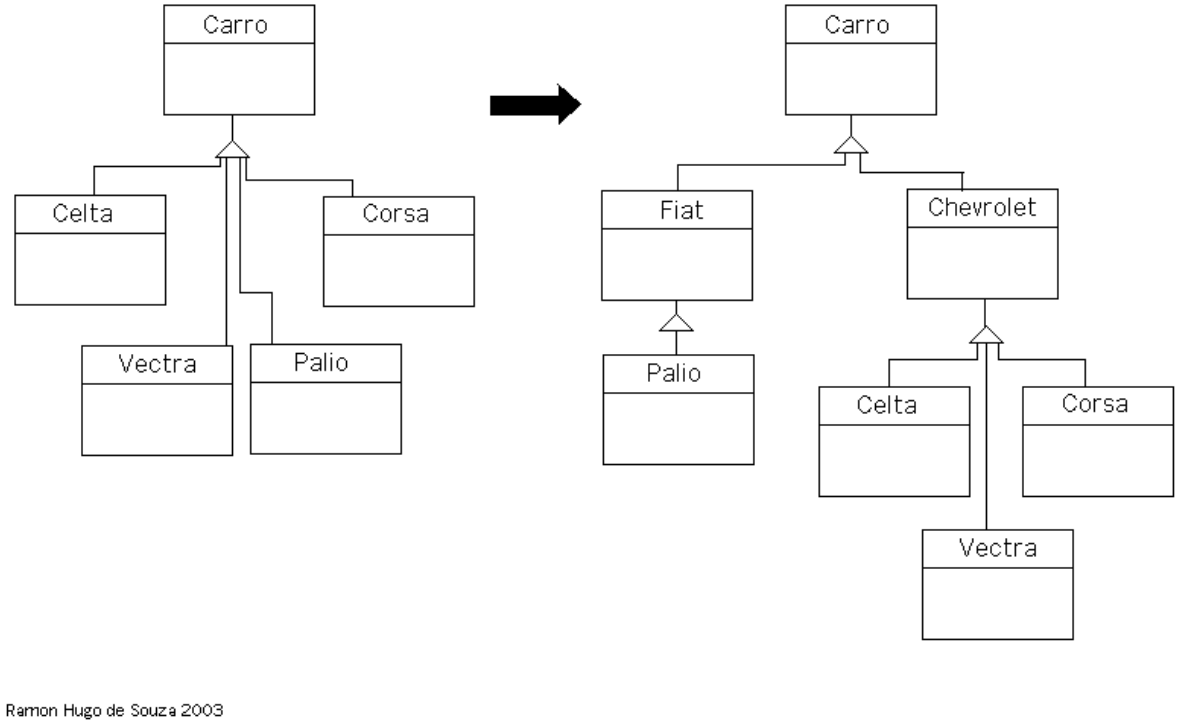


Figura 2 - Exemplo de refatoração.

O exemplo é trivial, mas poderíamos, por exemplo, manter todos os tipos mencionados de carros como classes filhas de Carro, e criar uma nova superclasse Conversíveis, no caso de existência, por exemplo, de um carro conversível.

2.2.3 Maus Cheiros

Maus cheiros são ocasiões em que o programador sente que alguma coisa está errada no código. Na maioria destes casos a aplicação de técnicas de refatoração resolve o problema.

No caso de código duplicado a refatoração pode, por exemplo, “retirar” o código duplicado dos métodos que os contenham e criar um novo método de uso comum a esses dois métodos.

Se os problemas forem métodos muito longos a refatoração pode dividir um método em várias sub-funcionalidades do mesmo, tornando o código mais fácil de entender e mais reutilizável.

No caso de não se ter certeza sobre generalidade de métodos ou atributos pode ser utilizada a refatoração para verificação da necessidade de criação de superclasses até então não existentes.

Enfim, existem muitos outros casos em que pode ser sentida a necessidade do uso de técnicas de refatoração.

Capítulo 3 – Projeto e Resultados Experimentais

3.1 Especificação do Sistema

A complexidade dos sistemas computacionais desenvolvidos hoje em dia levou ao desenvolvimento de várias técnicas para sua devida especificação de forma a facilitar o desenvolvimento destes. Muitas destas formas se desenvolveram em complexos estudos de como se desenvolver um software por etapas bem definidas ou mesmo por blocos de etapas iterativas, podendo estas depois de analisadas serem repassadas.

As tecnologias utilizadas neste estudo não permitem possibilidades muito grandes no que se refere a uma modelagem. Quanto aos dispositivos em que se pretende demonstrar os dados temos dispositivos móveis com tecnologia Java, que possui uma API reduzida da API Java. Devido a estes fatores e também por preferência do autor foi escolhida a abordagem de programação extrema no que se refere à modelagem e desenvolvimento do protótipo do protótipo do sistema denominado de Preguiça.

3.1.1 Prototipação

Prototipação segundo [PRESSMAN 1995] é descrita em paradigmas de engenharia de software como uma das formas clássicas para se desenvolver sistemas quando da não possibilidade de obtenção absoluta dos dados com o cliente em questão. Quando esta abordagem é utilizada no desenvolvimento de sistemas é recomendável

que se explicite de forma bem clara que o protótipo é uma coisa totalmente independente do produto final.

A abordagem utilizada no sistema Preguiça não é a abordagem clássica de engenharia de software, afinal programação extrema poderia partir de um protótipo e em ciclos fazer com que este chegasse a se tornar o produto final, sendo sempre avaliado de forma dinâmica pelo usuário final. Porém um primeiro protótipo independente do sistema final foi desenvolvido para se ter idéia dos requisitos iniciais e uma idéia de como se modelar o futuro banco de dados no qual o sistema irá basear suas pesquisas.

As idéias do paradigma clássico de prototipação são mantidas quando da idéia de se desenvolver um programa separado do sistema para se ter uma noção de como seria o produto final – mas apenas nesta primeira etapa, e também a interação com o usuário para uma coleta de requisitos mais apurada. Porém os ciclos irão se manter desta maneira mesmo quando o protótipo for validade e se partir para o sistema final.
Esta escolha se deu pela facilidade de se utilizar tecnologias previamente conhecidas como programação para estações de trabalho e em seguida se partir para o sistema que seria na verdade desenvolvido para dispositivos móveis.

3.1.2 Especificando o Preguiça

Como o sistema Preguiça nasceu de necessidades de usuários de uma rede de transporte público, o que vem a incluir o autor do referido sistema, então se torna bem simples a interação entre o desenvolvedor do sistema e o usuário final, afinal de contas ambos vem a ser a mesma pessoa, porém as necessidades do usuário estudado se baseiam também em necessidades de outros usuários deste sistema de transporte público – estes consultados durante o processo de desenvolvimento.

3.1.2.1 Programa para levantamento de requisitos

O primeiro passo na construção do sistema veio a ser um ambiente para se verificar as necessidades do usuário quanto às informações que ele requisita em um sistema de horários de transporte público.

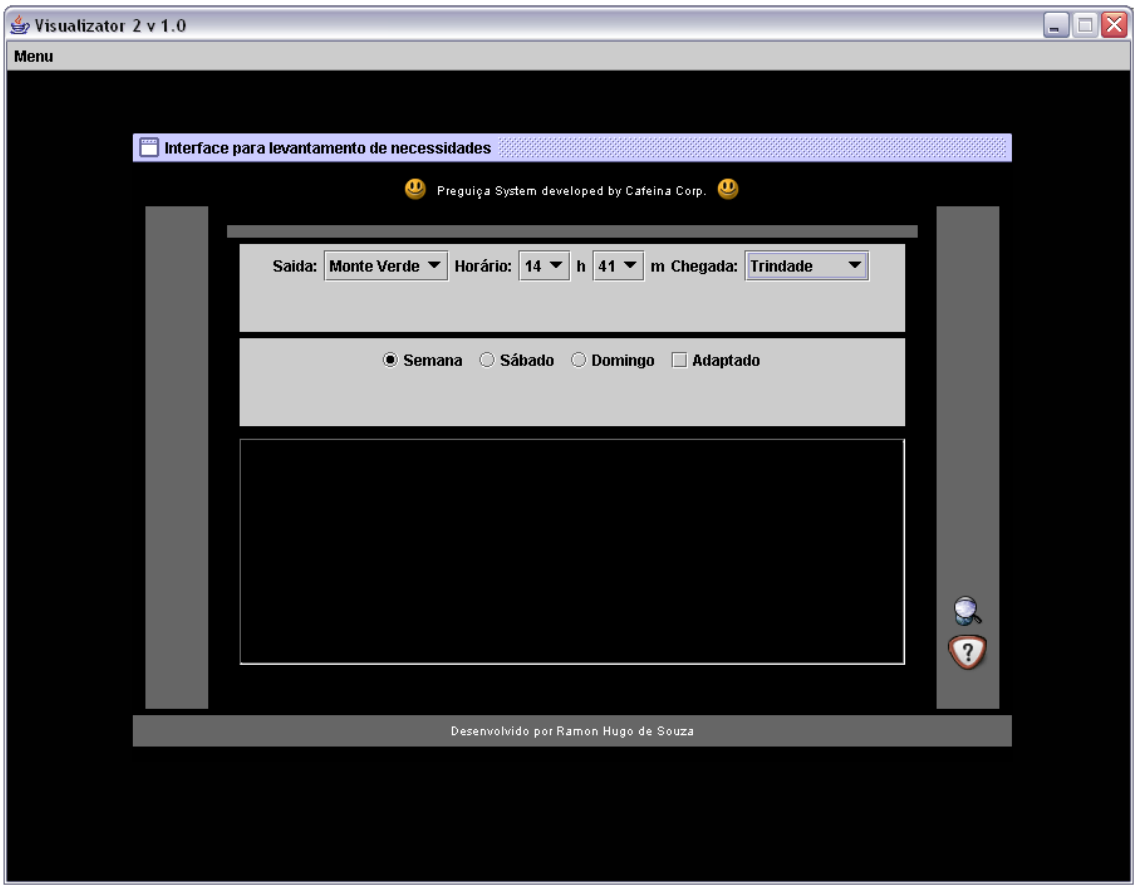


Figura 3 - Primeira versão do programa para levantar requisitos.

Por conveniência este sistema de levantamento de requisitos foi desenvolvido como uma aplicação para desktop e não para dispositivos móveis. Como a idéia de se criar um sistema destes era apenas para obter dados sobre a necessidade de clientes não foi preciso se preocupar com a tecnologia utilizada no seu desenvolvimento.

No primeiro modelo de levantamento de requisitos levou-se em consideração informações sobre de onde o usuário estava saindo, em que horário ele gostaria de sair, sendo o horário por padrão carregado com o horário do sistema, em que local ele gostaria de chegar, informações sobre o dia e se há a necessidade de se procurar por um ônibus com adaptação especial para deficientes físicos.

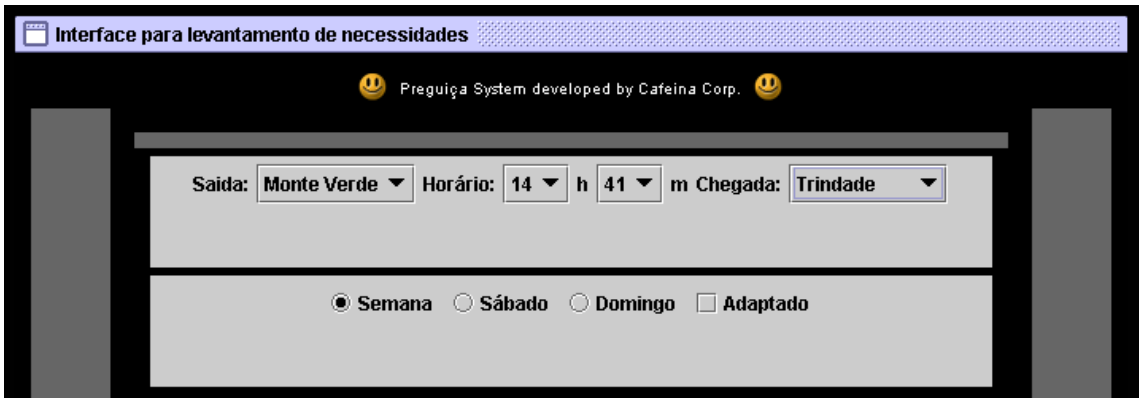


Figura 4 - Informações consideradas no primeiro modelo de levantamento de requisitos.

Aparentemente as necessidades de um usuário de um sistema de transporte público no sistema considerado se restringem apenas a estes poucos requisitos. Apesar de o banco de dados ser modelado de forma a possibilitar outras informações, o acesso do usuário considerado se restringe a estes dados.

As outras funcionalidades no sistema original eram um botão de help que explicava a finalidade do programa e um botão para iniciar a procura.

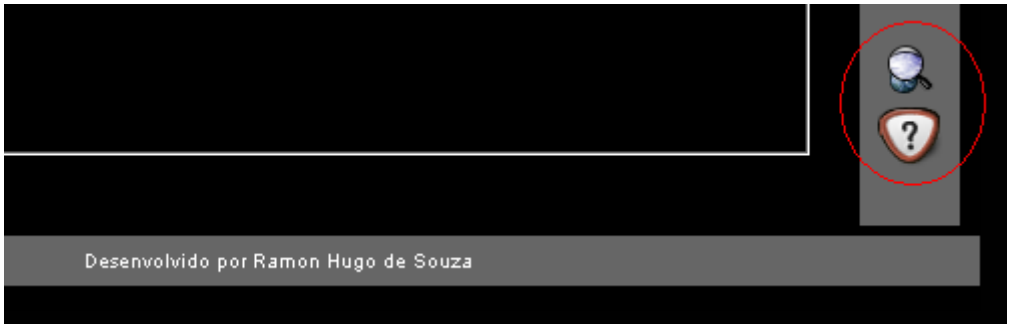


Figura 5 - Botões do modelo de levantamento de requisitos.

Ao se clicar no botão de ajuda obtinha-se a seguinte mensagem:

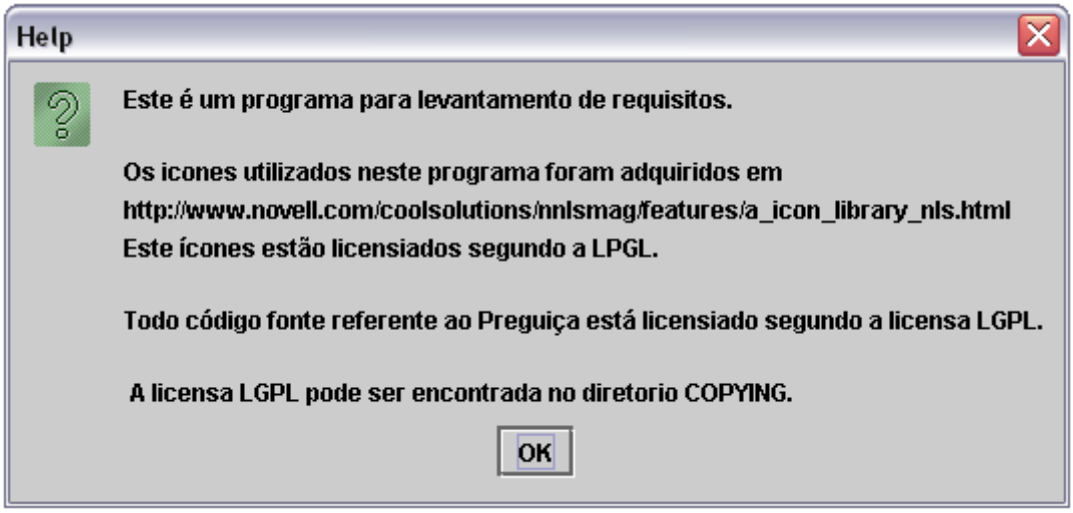


Figura 6 - Mensagem de help do programa de levantamento de requisitos.

Não houve preocupação quanto ao help específico do programa já que não seria este o programa final a ser utilizado pelo usuário. Como talvez viesse a ser utilizado por outros desenvolvedores foram colocadas informações referentes ao sistema – estas modificadas mais tarde.

Quando uma pesquisa era iniciada – ainda no primeiro modelo para avaliação das necessidades do usuário – o que lhe era retornado era apenas um aviso de inicio de procura e um aviso de que se trata de apenas um sistema para levantamento de necessidades de um usuário.

Figura 7 - Resultado da interação de pesquisa do primeiro modelo de levantamento de requisitos.

A próxima etapa seria então a criação de um banco de dados para que o programa pudesse acessar e então devolver ao usuário uma resposta a sua requisição. Considerou-se inicialmente o retorno de uma resposta aleatória porém baseada no banco de dados para esta etapa.

3.1.2.2 Modelo inicial do banco de dados

Do programa para levantamento de requisitos foram obtidas as informações de que seria necessário se considerar um local de saída e um local de chegada bem como um horário de saída.

Como se tratava de apenas uma modelagem inicial não era crítica a total coerência do banco em se tratando da necessidade do usuário, afinal ainda estava sendo considerado que novas necessidades iriam surgir nesta fase de levantamento dos primeiros requisitos.

O modelo de entidades e relacionamentos foi utilizado para uma visualização de como viria a ser o banco e foi o modelo adotado para se trabalhar com os conceitos relacionados com o banco de dados.

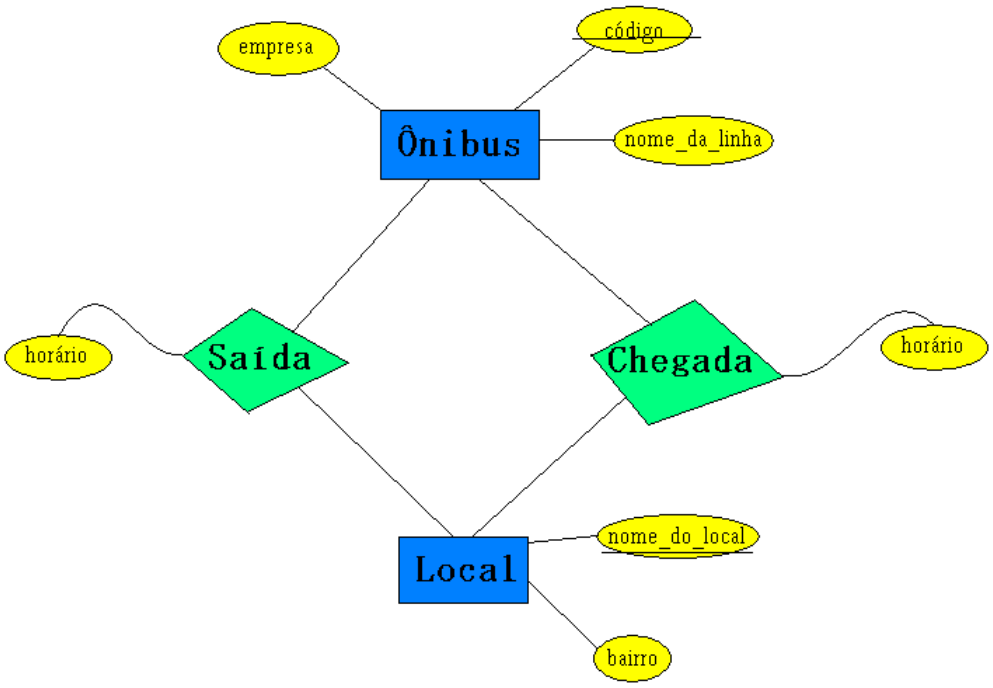


Figura 8 - Primeiro modelo de entidades e relacionamentos do banco de dados.

Alguns dados a mais foram considerados para uma possível utilização posterior, como o código do ônibus e uma empresa relacionada a um ônibus – porém esta ainda era uma versão bem simplificada do modelo de banco de dados.

Neste modelo notou-se a falta de uma tarifa relacionada com o ônibus, o que poderia ser posteriormente utilizado para busca de melhor caminho já que também se considerou a possível utilização de mais de uma empresa prestadora deste serviço. O local também ficaria mais bem modelado com um atributo que indicasse se ele seria um ponto de referencia no bairro ou mesmo uma sub-estação para mudança de linha.

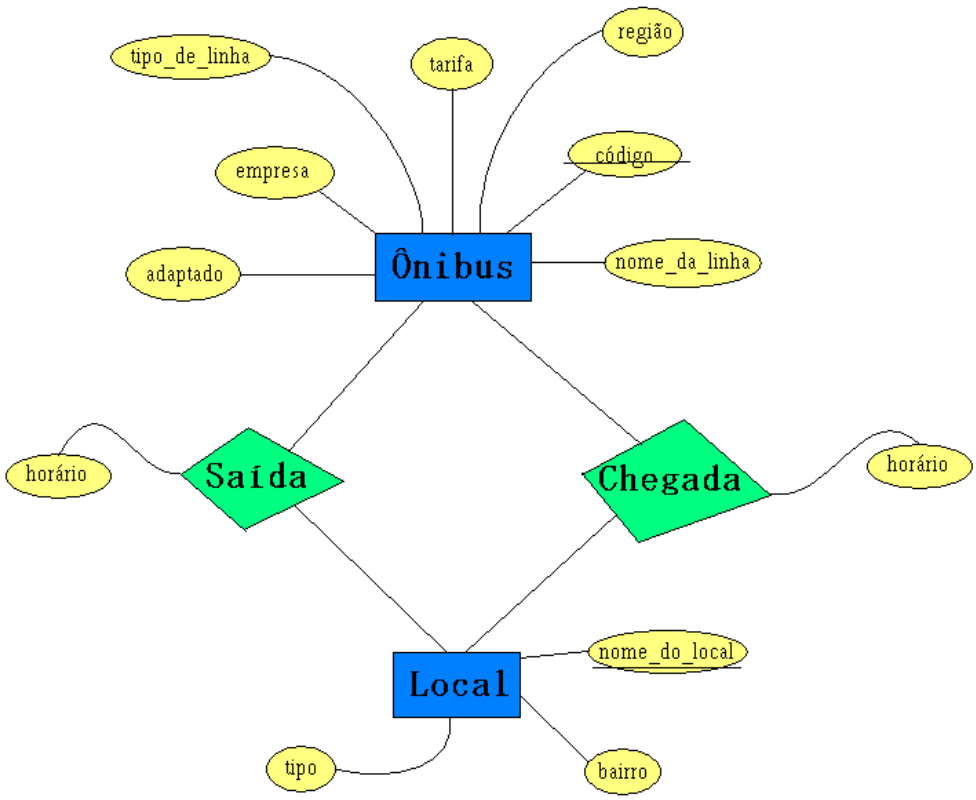


Figura 9 - Segundo modelo de entidades e relacionamentos do banco de dados.

As mudanças do modelo do banco nesta etapa não atrapalharam de modo algum o desenvolvimento do sistema, pois ainda estava-se trabalhando sobre as necessidades do cliente e não existia um banco de dados com informações reais neste ponto.

Alguns atributos foram adicionados devido ao sistema se basear em serviços de transporte público reais, e foram obtidos com as devidas fontes relacionadas às prestadoras do serviço. Estes dados não estão diretamente relacionados com a resolução do problema, mas pareceu uma boa idéia incluí-los na modelagem para um possível uso posterior.

A partir deste modelo começou-se a perceber que algo não estava realmente bem definido neste modelo de um banco de dados. Como, por exemplo, o fato do ônibus ser adaptado. Como o ônibus neste caso se referia a uma linha e não a um ônibus específico parecia que seria melhor relacionar a adaptação a um horário de modo que se resolveu colocar a informação sobre adaptação nos relacionamentos de saída e chegada.

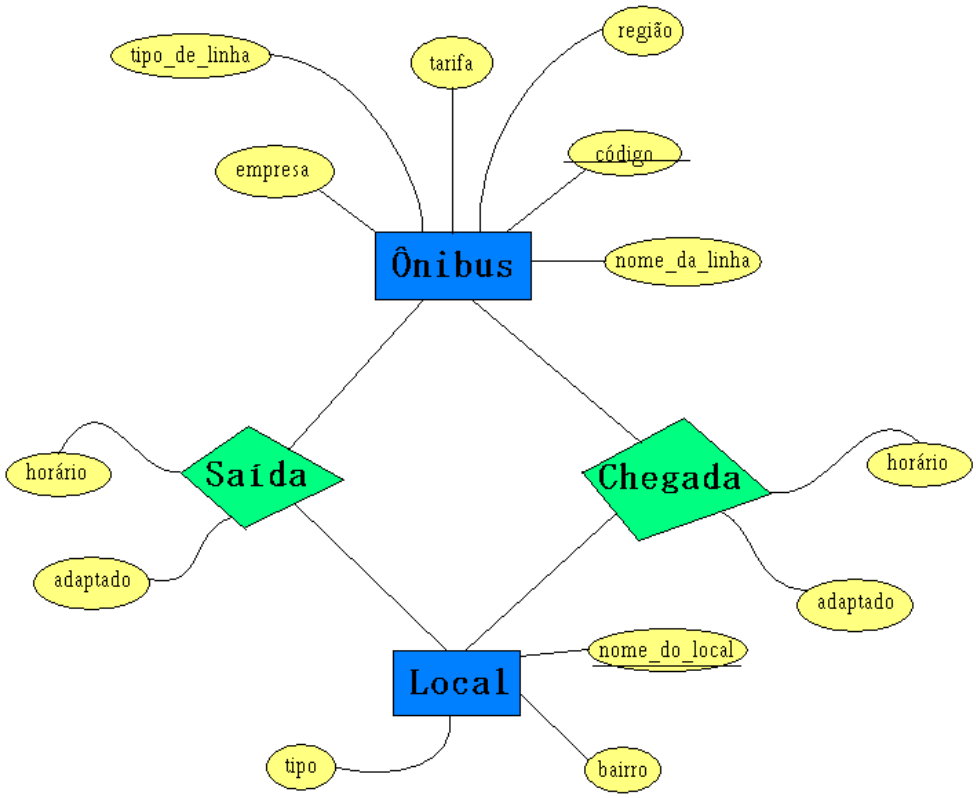


Figura 10 - Terceiro modelo de entidades e relacionamentos do banco de dados.

A partir do terceiro modelo do banco começou-se a analisar os dados que viriam a ser os dados utilizados para a resolução do problema de busca de melhor caminho. O modelo do banco ainda não estava completo e sabia-se que possivelmente ainda possuía erros na sua definição. Um deles sendo bastante obvio, como o fato de que o que chamamos de ônibus só poderia ser uma entidade que se relacionaria como um “isa” com o que viria realmente a ser o ônibus, mas já chegaremos neste ponto.

Partindo-se então para uma primeira olhada nos dados reais sobre os ônibus percebeu-se mais uma barreira a ser transposta, o fato de que os dados não estavam diretamente disponíveis. Eles poderiam ser acessados através de uma página html, porém não se teria acesso direto aos dados.

Como as entradas de dados eram muitas – mesmo considerando-se apenas onze linhas de ônibus – foi feito um programa em c – Anexo 1 – para extrair estes dados. O que facilitou bastante a criação de um banco de dados experimental.

O referido programa em c lia como entrada um arquivo que resultava da copia textual dos dados exibidos em html e retornava um arquivo para a criação do banco em mySQL – que foi o banco de dados escolhido para abrigar os dados referentes ao modelo de levantamento de requisitos.

As entradas para o banco de dados eram realmente muitas, e como o banco viria a ser modificado com certeza ainda neste ponto não se cogitava a possibilidade de se perder tempo passando-se dados manualmente.

As saídas em mySQL se basearam no formato de dados resultado da mudança do paradigma entidade relacionamento para o paradigma relacional resultando nestas definições para o banco:

CREATE TABLE onibus (nome_empresa char(15), codigo_onibus char(5) not null, nome_da_linha char(20), tarifa char(5), tipo_de_linha char(10), regiao char(7), PRIMARY KEY(codigo_onibus));

CREATE TABLE local (nome_do_local char(10) not null, bairro char(20), tipo char(15), PRIMARY KEY(nome_do_local));

CREATE TABLE saida (codigo_onibus char(5) not null, nome_do_local char(10) not null, horario char(10), adaptado char(3), PRIMARY KEY(codigo_onibus, nome_do_local));

CREATE TABLE chegada (codigo_onibus char(5) not null, nome_do_local char(10) not null, horario char(10), adaptado char(3), PRIMARY KEY(codigo_onibus, nome_do_local));

Com esse primeiro modelo em um banco de dados real notou-se que faltavam ainda informações sobre os dias de circulação do ônibus e que o horário dos referidos ônibus deveriam fazer parte da chave dos relacionamentos de saída e chegada. Sendo

essas falhas corrigidas nos dois modelos subseqüentes, e então se chegou no quinto modelo do banco dos dados – o quarto modelo foi integrado diretamente no quinto.

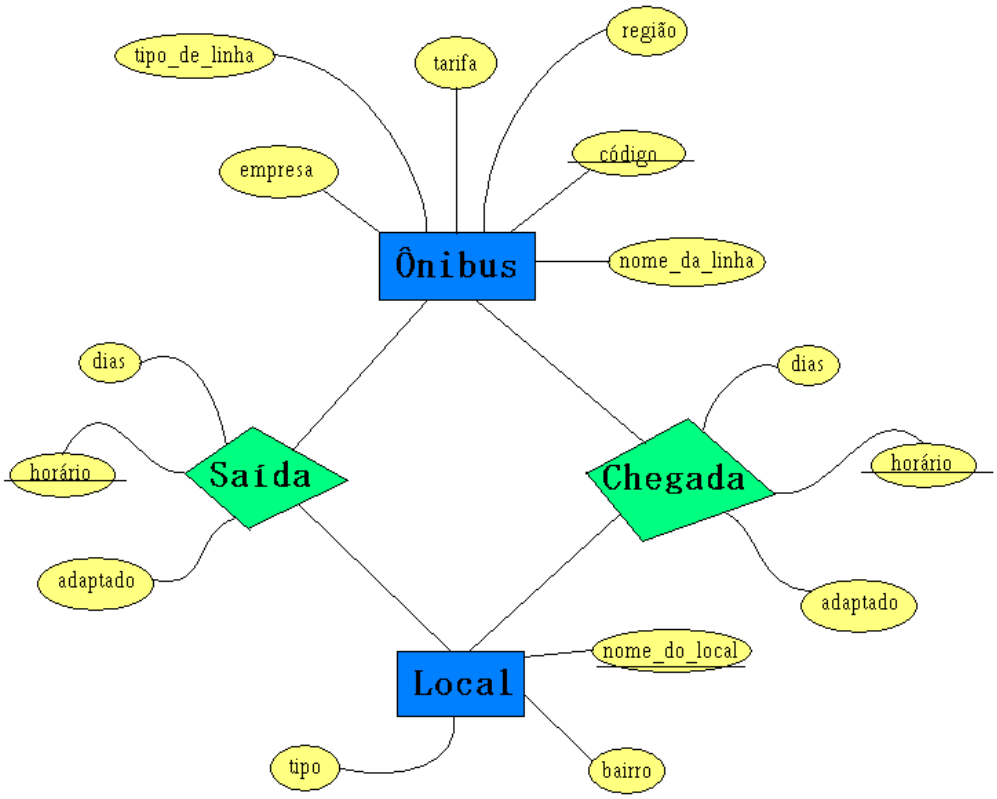


Figura 11 - Quinto modelo de entidades e relacionamentos do banco de dados.

Neste modelo então se percebeu o maior erro desta modelagem – o fato de que ônibus é na verdade uma linha, e não um ônibus individual. O que torna o banco de dados totalmente inválido, já que não se torna possível fazer nenhuma pesquisa no mesmo.

O banco foi modelado pensando-se em uma otimização para evitar redundância de dados, o que acabou levando a uma indução errônea sobre como modelá-lo. Ao se verificar na prática com dados reais pode-se perceber que não haveria como se realizar consultas com esse modelo.

Apesar de todo o invés que surgiu com a percepção deste erro isto foi facilmente corrigido transformando o que seria uma linha em um ônibus em si. Adicionou-se então na entidade ônibus o que chamamos de id e assim individualizamos cada instância de uma linha. Poderia-se considerar ônibus e linhas como entidades separadas com um relacionamento entre elas, porém resolveu-se somente adicionar o atributo id em ônibus nesta fase de modelagem, mas na criação do banco a redundância foi corrigida.

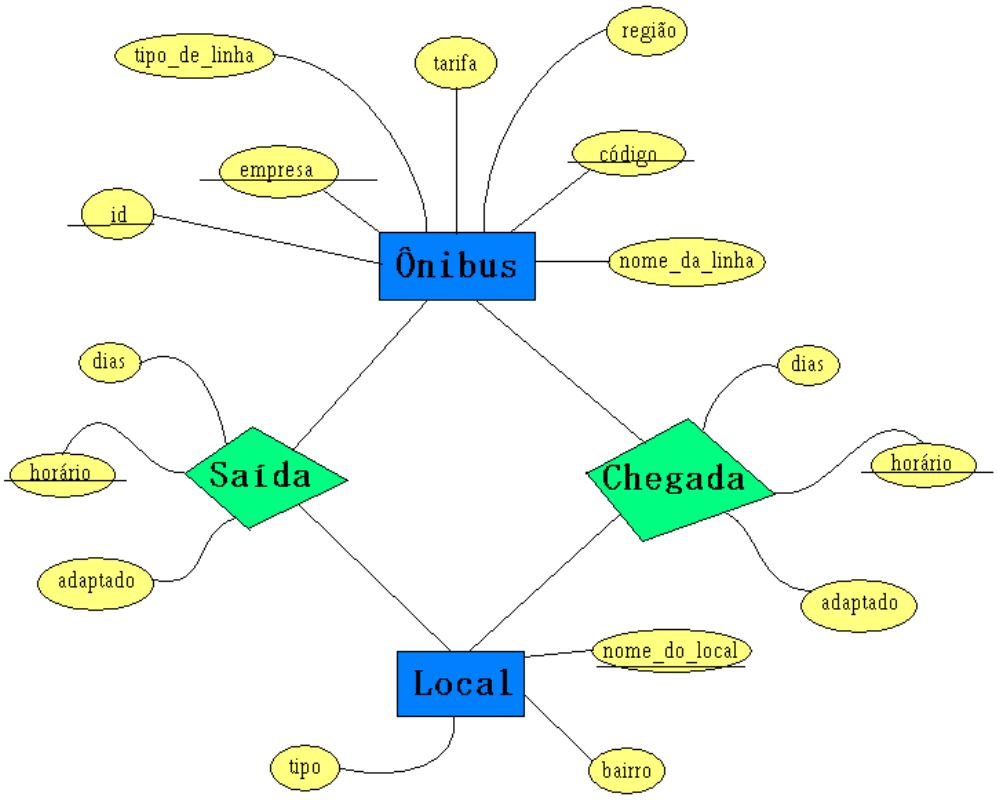


Figura 12 - Sexto modelo de entidades e relacionamentos do banco de dados.

Chegamos então ao modelo que seria utilizado na próxima etapa relativa aos requisitos do sistema. O código testado em mySQL deste modelo foi o seguinte:

CREATE TABLE linha (

```
        codigo_onibus MEDIUMINT(5) UNSIGNED ZEROFILL DEFAULT '00000' NOT NULL,
        nome_empresa CHAR(15) DEFAULT '' NOT NULL,
        nome_da_linha CHAR(30) DEFAULT '' NOT NULL,
        tarifa FLOAT(2,2) ZEROFILL DEFAULT '00.00' NOT NULL,
tipo_de_linha TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL
        regiao SMALLINT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
                PRIMARY KEY(codigo_onibus)
        );

CREATE TABLE onibus (
        id_onibus SMALLINT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
codigo_onibus MEDIUMINT(5) UNSIGNED ZEROFILL DEFAULT '00000' NOT NULL, REFERENCES Linha
        PRIMARY KEY(id_onibus, codigo_onibus)
);

CREATE TABLE local (
        nome_do_local CHAR(20) DEFAULT '' NOT NULL,
        bairro CHAR(20) DEFAULT '' NOT NULL,
        tipo TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL,
        PRIMARY KEY(nome_do_local)
        );

CREATE TABLE saida (
        id_onibus SMALLINT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL REFERENCES onibus,
        codigo_onibus INT(5) UNSIGNED DEFAULT '00000' NOT NULL REFERENCES onibus,
        nome_do_local CHAR(10) DEFAULT '' NOT NULL REFERENCES local,

        dias TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL,
```

```
    horario TIME DEFAULT '00:00:00' NOT NULL,
        adaptado  TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL,
            PRIMARY KEY(id_onibus, codigo_onibus, nome_do_local, dias,
horario)
    );

CREATE TABLE chegada (
    id_onibus SMALLINT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT
NULL REFERENCES onibus,
    codigo_onibus INT(5) UNSIGNED DEFAULT '00000' NOT NULL
REFERENCES onibus,
    nome_do_local CHAR(10) DEFAULT '' NOT NULL REFERENCES local,

    dias TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL,

    horario TIME DEFAULT '00:00:00' NOT NULL,
    adaptado  TINYINT(1) UNSIGNED DEFAULT '0' NOT NULL,
    PRIMARY KEY(id_onibus, codigo_onibus, nome_do_local, dias, horario)

    );
```

Durante o mapeamento para este primeiro banco exemplo resolvemos então considerar um relacionamento do tipo Isa entre linha e ônibus, sendo que o que chamamos inicialmente de ônibus veio enfim a ser uma linha e suas “instâncias” os ônibus.

3.1.2.3 Criando um banco de dados real para o programa de levantamento de requisitos

Para a próxima etapa de levantamento de requisitos do usuário foi considerado o acesso a dados em um banco especificado. Foi utilizado o banco de dados servido pelo

departamento de informática e estatística da UFSC, e para que fosse possível a utilização deste banco o modelo definido em mySQL não foi utilizado diretamente, já que as tabelas foram colocadas no banco através de um administrador de dados do sistema utilizado, porém é o mesmo modelo, exceto pelo fato de ter-se utilizado varchar para variáveis relativas a palavras.

Após serem passados os dados para o banco real ficou-se com as seguintes tabelas:

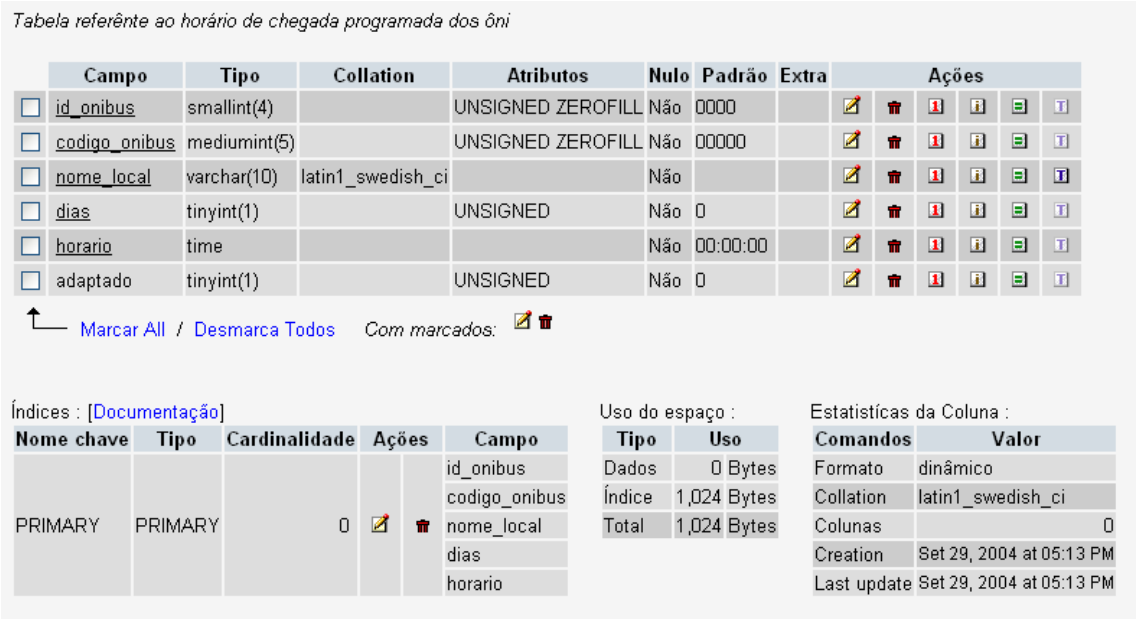


Tabela referênte ao horário de chegada programada dos ôni

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
☐	id_onibus	smallint(4)		UNSIGNED ZEROFILL	Não	0000		
☐	codigo_onibus	mediumint(5)		UNSIGNED ZEROFILL	Não	00000		
☐	nome_local	varchar(10)	latin1_swedish_ci		Não			
☐	dias	tinyint(1)		UNSIGNED	Não	0		
☐	horario	time			Não	00:00:00		
☐	adaptado	tinyint(1)		UNSIGNED	Não	0		

Marcar All / Desmarca Todos *Com marcados:*

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações		Campo
PRIMARY	PRIMARY	0			id_onibus
					codigo_onibus
					nome_local
					dias
					horario

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	dinâmico
Collation	latin1_swedish_ci
Colunas	0
Creation	Set 29, 2004 at 05:13 PM
Last update	Set 29, 2004 at 05:13 PM

Figura 13- Tabela chegada no banco de dados mySQL.

Tabela referente à linhas de ônibus

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
☐	codigo_onibus	mediumint(5)		UNSIGNED ZEROFILL	Não	00000		
☐	nome_empresa	varchar(15)	latin1_swedish_ci		Não			
☐	nome_linha	varchar(30)	latin1_swedish_ci		Não			
☐	tarifa	float(3,2)		UNSIGNED ZEROFILL	Não	0.00		
☐	tipo_linha	tinyint(1)		UNSIGNED	Não	0		
☐	regiao	smallint(4)		UNSIGNED ZEROFILL	Não	0000		

Marcar All / Desmarca Todos *Com marcados:*

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		codigo_onibus

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	dinâmico
Collation	latin1_swedish_ci
Colunas	0
Creation	Set 29, 2004 at 04:58 PM
Last update	Set 29, 2004 at 04:58 PM

Figura 14 - Tabela linha no banco de dados mySQL.

Tabela que define os locais de onde os ônibus podem sair e

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
☐	nome_local	varchar(20)	latin1_swedish_ci		Não			
☐	bairro	varchar(20)	latin1_swedish_ci		Não			
☐	tipo_local	tinyint(1)		UNSIGNED	Não	0		

Marcar All / Desmarca Todos *Com marcados:*

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		nome_local

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	dinâmico
Collation	latin1_swedish_ci
Colunas	0
Creation	Set 29, 2004 at 05:51 PM
Last update	Set 29, 2004 at 05:51 PM

Figura 15 - Tabela local no banco de dados mySQL.

Tabela que define instancias especificas de ônibus

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
☐	id_onibus	smallint(4)		UNSIGNED ZEROFILL	Não	0000		
☐	codigo_onibus	mediumint(5)		UNSIGNED ZEROFILL	Não	00000		

Marcar All / Desmarca Todos *Com marcados:*

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		id_onibus
				codigo_onibus

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	fixo
Collation	latin1_swedish_ci
Colunas	0
Creation	Set 29, 2004 at 05:05 PM
Last update	Set 29, 2004 at 05:05 PM

Figura 16 - Tabela onibus no banco de dados mySQL.

Tabela que define os horários de saída de ônibus

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
☐	id_onibus	smallint(4)		UNSIGNED ZEROFILL	Não	0000		
☐	codigo_onibus	int(5)		UNSIGNED ZEROFILL	Não	00000		
☐	nome_local	varchar(10)	latin1_swedish_ci		Não			
☐	dias	tinyint(1)		UNSIGNED	Não	0		
☐	horario	time			Não	00:00:00		
☐	adaptado	tinyint(1)		UNSIGNED	Não	0		

Marcar All / Desmarca Todos *Com marcados:*

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		id_onibus
				codigo_onibus
				nome_local
				dias
				horario

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	dinâmico
Collation	latin1_swedish_ci
Colunas	0
Creation	Set 29, 2004 at 05:09 PM
Last update	Set 29, 2004 at 05:09 PM

Figura 17 - Tabela saida no banco de dados mySQL.

Como os dados para este exemplo seriam obtidos pelo programa feito previamente em c que gera como saída código para gerar dados para estas tabelas definidas em mySQL teve-se que inserir previamente, por conveniência, os dados que não são automaticamente gerados com este programa. Foi exatamente por este motivo que definimos uma entidade ônibus e uma linha.

Para linha colocamos as seguintes entradas:

```
INSERT INTO linha VALUES('110', 'Transol', 'TICEN-TITRI Direto', '01.50', '2', '0001');
INSERT INTO linha VALUES('120', 'Transol', 'Gama D'Eça Semidireto', '01.50', '1', '0001');
INSERT INTO linha VALUES('121', 'Transol', 'Mauro Ramos Semidireto', '01.50', '1', '0001');
INSERT INTO linha VALUES('131', 'Transol', 'Agronômica via Gama D'Eça', '01.50', '0', '0001');
INSERT INTO linha VALUES('132', 'Transol', 'Agronomica via Gama D'eça/H.I.', '01.50', '0', '0001');
INSERT INTO linha VALUES('133', 'Transol', 'Agronômica via Mauro Ramos', '01.50', '0', '0001');
INSERT INTO linha VALUES('134', 'Transol', 'Beira-Mar Norte', '01.50', '0', '0001');
INSERT INTO linha VALUES('168', 'Transol', 'Monte Verde', '01.50', '0', '0001');
INSERT INTO linha VALUES('169', 'Transol', 'Monte Verde via Mané Vicente', '01.50', '0', '0001');
INSERT INTO linha VALUES('170', 'Transol', 'Caminho da Cruz via João Paulo', '01.50', '0', '0001');
INSERT INTO linha VALUES('176', 'Transol', 'Saco Grande via H.U.', '01.50', '0', '0001');
```

Para local colocamos as seguintes entradas:

```
INSERT INTO local VALUES('TICEN', 'Centro', '1');
INSERT INTO local VALUES('TITRI', 'Trindade', '1');
INSERT INTO local VALUES('SM Florêncio', 'Monte Verde', '0');
INSERT INTO local VALUES('Posto Sambaqui', 'Sambaqui', '1');
```

Como para chegada e saída são gerados alguns milhares de tuplas baseadas em dados reais, fica facilmente explicada a necessidade do programa para extração de dados.

A tabela linha quando povoada ficou do seguinte modo:

←T→	codigo_onibus	nome_empresa	nome_linha	tarifa	tipo_linha	regiao
	00110	transol	ticen-titri	1.50	2	0001
	00120	transol	gama deça semidireto	1.50	1	0001
	00121	transol	mauro ramos semidireto	1.50	1	0001
	00131	transol	agronomica via gama deça	1.50	0	0001
	00132	transol	agronomica via gama deca/hi	1.50	0	0001
	00133	transol	agronomica via mauro ramos	1.50	0	0001
	00134	transol	beira-mar norte	1.50	0	0001
	00168	transol	monte verde	1.50	0	0001
	00169	transol	monte verde via mane vicente	1.50	0	0001
	00170	transol	caminho da cruz via joao paulo	1.50	0	0001
	00176	transol	saco grande	1.50	0	0001

Com marcados:

Figura 18 - Tabela linha povoada.

A tabela local quando povoada ficou do seguinte modo:

←T→	nome_local	bairro	tipo_local
	ticen	centro	1
	titri	trindade	1
	sm florencio	monte verde	0
	posto sambaqui	sambaqui	1

Com marcados:

Figura 19 - Tabela local povoada.

Percebeu-se então que seria mais simples se os campos id_onibus nas tabelas ônibus, saída e chegada fossem auto incrementados.

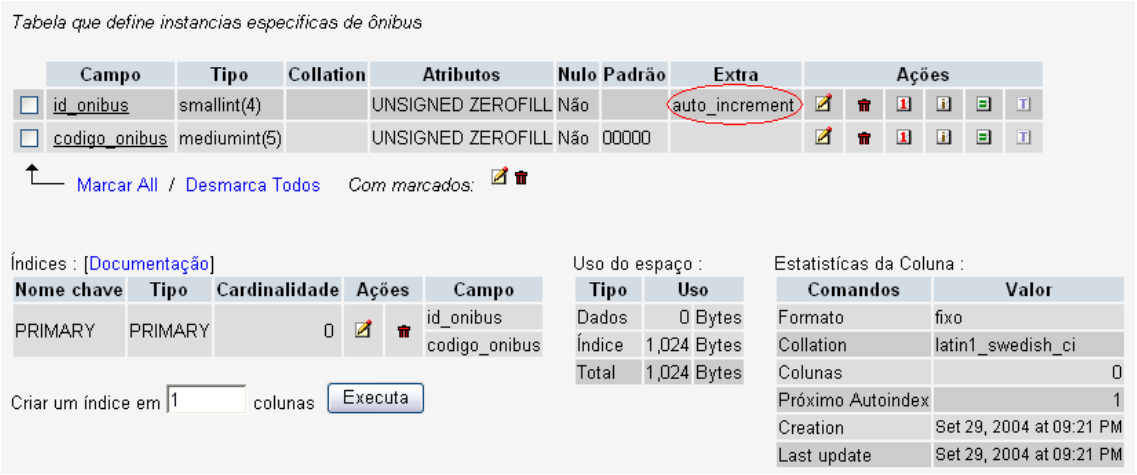


Tabela que define instancias especificas de ônibus

Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
id_onibus	smallint(4)		UNSIGNED ZEROFILL	Não		auto_increment	
codigo_onibus	mediumint(5)		UNSIGNED ZEROFILL	Não	00000		

Marcar All / Desmarca Todos Com marcados:

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		id_onibus
				codigo_onibus

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatistícas da Coluna :

Comandos	Valor
Formato	fixo
Collation	latin1_swedish_ci
Colunas	0
Próximo Autoindex	1
Creation	Set 29, 2004 at 09:21 PM
Last update	Set 29, 2004 at 09:21 PM

Figura 20 - Tabela ônibus já com o auto-incremento.

As tabelas referentes a ônibus, saída e linha não são aqui demonstradas devido ao seu extenso número de tuplas.

O total de ônibus – que na verdade seriam horários de linhas – computados nas 11 linhas foi de 1734. O que em uma abordagem “normal” por ser um problema np-completo nos daria 3006756 combinações para o caso de uma mudança de linha possível em uma estação. É claro que com algumas seleções previas teríamos

aproximadamente 187922 combinações – isso no caso de estarmos considerando que existem 4 locais possíveis então considerar-se-ia q a saída teria ¼ do valor de linhas e a chegada outro ¼ do valor das linhas. E ainda poder-se-ia descontar horários de sábados e domingo para o caso de pesquisas na semana – por exemplo. Porém ainda ter-se-ia que comparar as saídas entre si para se achar a melhor solução.

Tendo agora um banco de dados real o próximo passo é conectá-lo com o programa de levantamento de requisitos para verificar se as pesquisas como desejamos são possíveis de serem realizadas.

3.1.2.4 Conectando o banco de dados de teste com o programa de levantamento de requisitos.

O primeiro objetivo era simplesmente realizar uma consulta em mySQL sem preocupações referentes ao banco de dados. Se o banco se demonstrasse consistente em relação aos dados necessários seria utilizado.

O banco de dados da rede inf só pode ser acessado por computadores internos a rede, então para solucionar o problema foi digitado no prompt de comando o seguinte:

```
ssh2 -L 3306:mysql.inf.ufsc.br:3306 ramonh@venus.inf.ufsc.br
```

Assim foi feito um tunelamento na porta 3306 e trabalhou-se como se o banco de dados estivesse sendo servido na rede local.

Novamente surgiram problemas referentes ao banco de dados quando da primeira pesquisa executada. Um erro foi referente à implementação das informações referentes à especificação de nome_local em saída e chegada, que estava definida como 10 em vez de 20 caracteres. Essas duas tabelas tiveram seus dados extraídos diretamente das paginas html da empresa em questão e ouve também um problema referente ao banco estar em linux e ser case sensitive. A definição das tabelas foi arrumada e em seguida os dados foram re-inseridos.

A interface de levantamento de requisitos foi modificada adicionando-se o campo “variação”, que se referia a uma variação em minutos do horário de saída e foi adaptada para mostrar resultados de pesquisa em banco de dados.

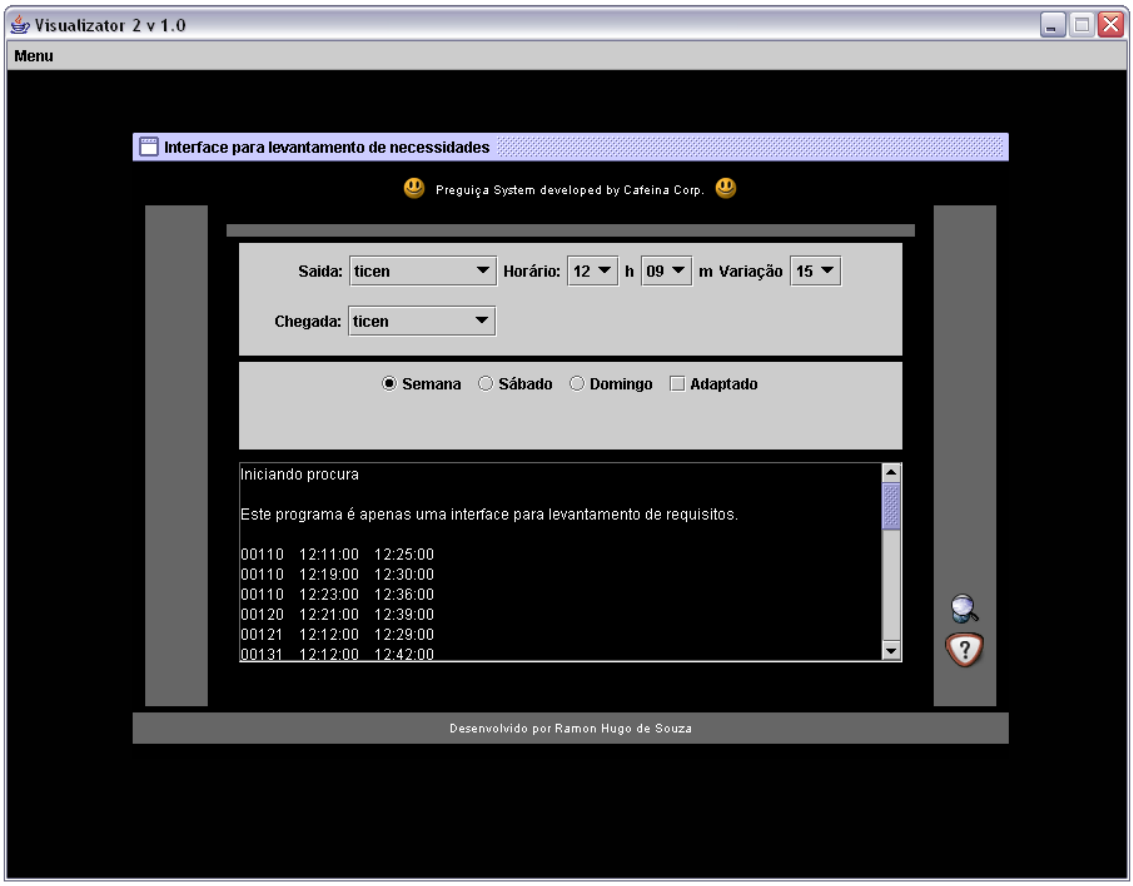


Figura 21 - Segunda versão do programa de levantamento de requisitos.

Para se testar a validade deste campo foi realizada a seguinte pesquisa SQL:

SELECT s.codigo_onibus, s.horario, c.horario FROM saida s, chegada c WHERE s.id_onibus = c.id_onibus and s.horario > 'horarioSaida' and s.horario < 'horarioSaidaLimite'

Onde horárioSaida é o horário selecionado pelo usuário e horarioSaidaLimite este horário somado com a variação também definida pelo usuário. Não foi selecionada nenhuma linha em específico, o que permitiu a verificação das respostas indepentemente da linha. Como muitas vezes os horários variavam muito, especialmente em sábados e domingos optou-se por não mais se utilizar o campo de variação, deixando para uma verificação futura.

Feita uma pré-seleção das entradas o primeiro susto. O protótipo resolvia as soluções de melhor caminho em tempo de 5 a 12 segundos apenas com acesso ao banco de dados.

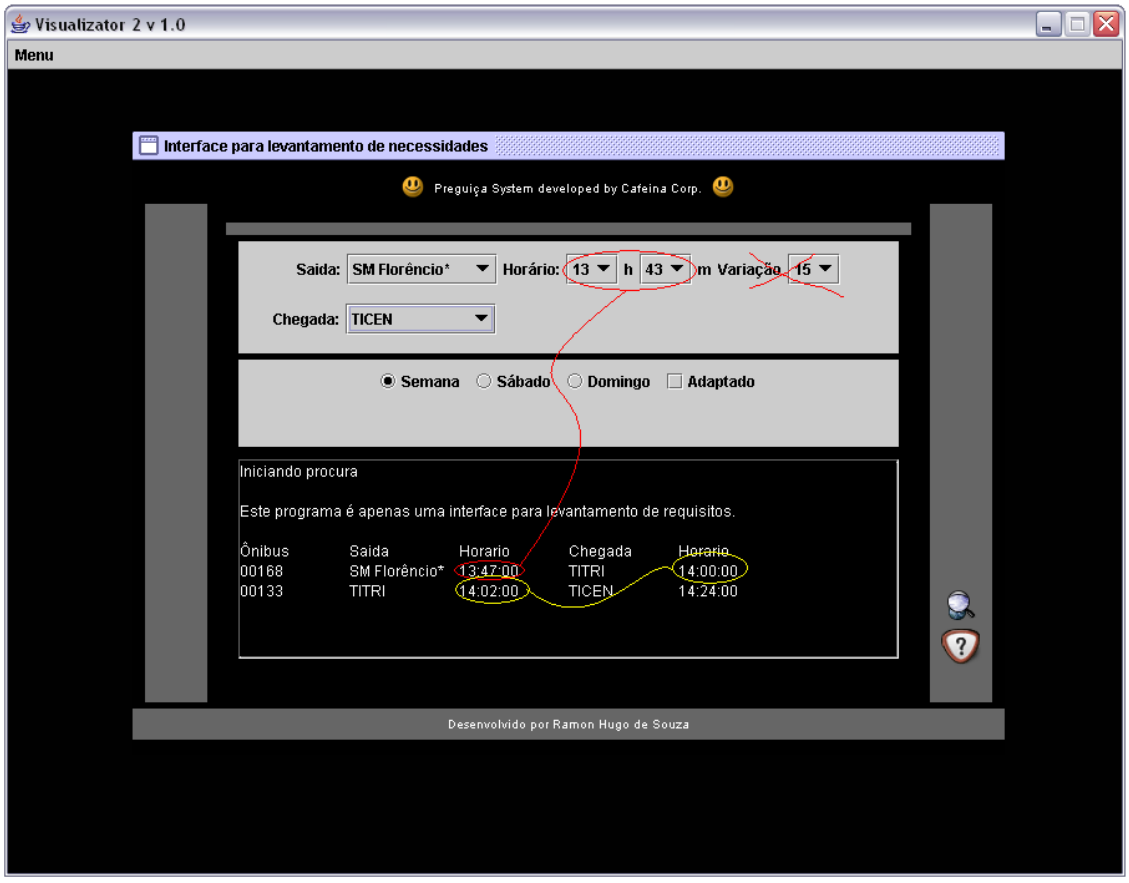


Figura 22 - Terceira versão do programa de levantamento de requisitos.

É claro que era uma procura mais simplificada em relação ao problema original, afinal não levava em consideração os bairros por onde o ônibus passava, e sim apenas pontos de referência.

Assim validou-se o banco de dados modelado quanto a essa pesquisa inicial do modelo, porém as necessidades do cliente não haviam sido totalmente supridas ainda, afinal não havia a possibilidade de calculo para bairros intermediários aos pontos. Por mais que não existisse informação sobre o horário que o ônibus passaria pelo bairro dever-se-ia de algum modo poder calcular um caminho que passasse por um bairro intermediário.

A solução para este novo problema estaria diretamente relacionada ao banco de dados. Era necessário que uma linha possuísse uma cadeia de bairros intermediários relacionada a ela – afinal saída e chagada já eram relacionamentos ligados a uma entidade local.

Ao se analisar as linhas concluiu-se que um dado intermediário era necessário a algumas linhas, já que estas saiam de um ponto e chegavam em outro, mas a sua finalidade era o local intermediário. Existiam também linhas alternativas que passavam por um local diferenciado – ou mais, e linhas que precisavam da localização segundo uma avenida principal. Em alguns casos era preciso referenciar um bairro intermediário e uma avenida. Assim surgiram duas novas características às linhas: “via” que determinaria a passagem por uma avenida ou local principal e “bairro intermediário” que determinaria um bairro por onde o ônibus passaria. O cliente ainda deveria poder especificar esses dados independentemente do fato de ser apenas uma avenida ou um bairro intermediário – para sua comodidade.

Foi decidido por uma cadeia intermediária relacionada a cada linha de ônibus, independente se era um bairro ou uma avenida.

Criou-se então um relacionamento denominado intermediários para interligar linha com local.

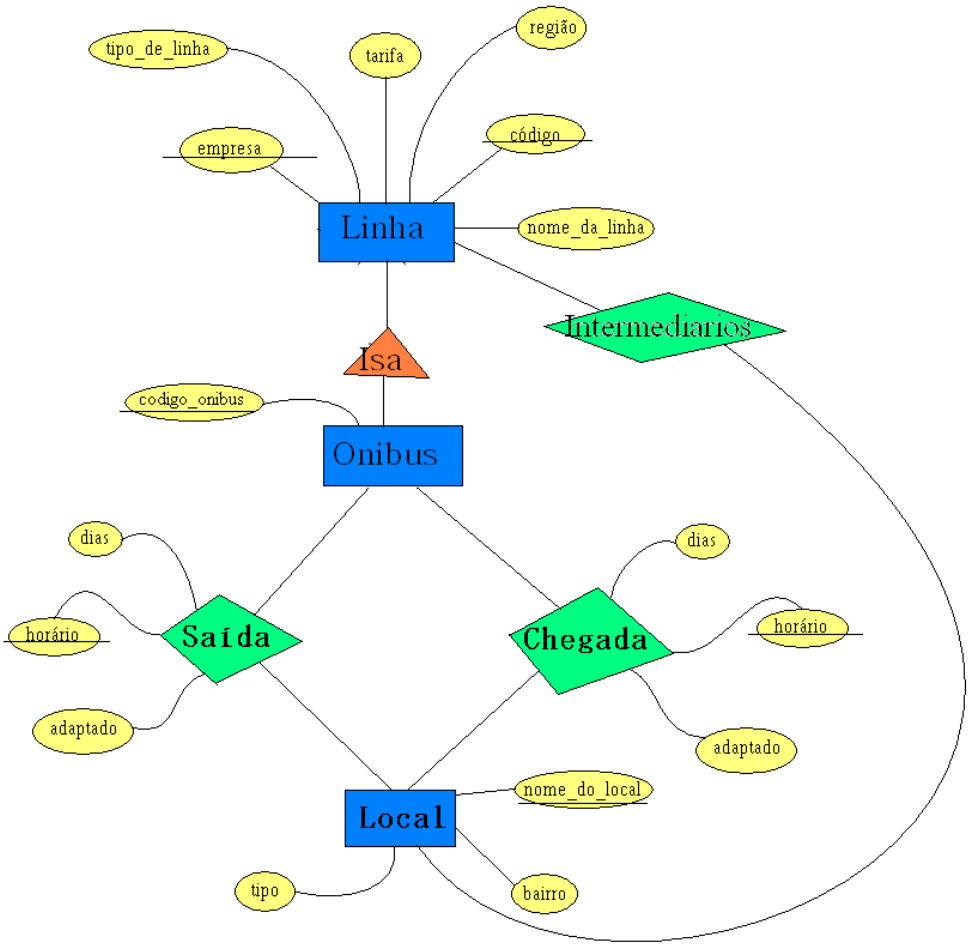
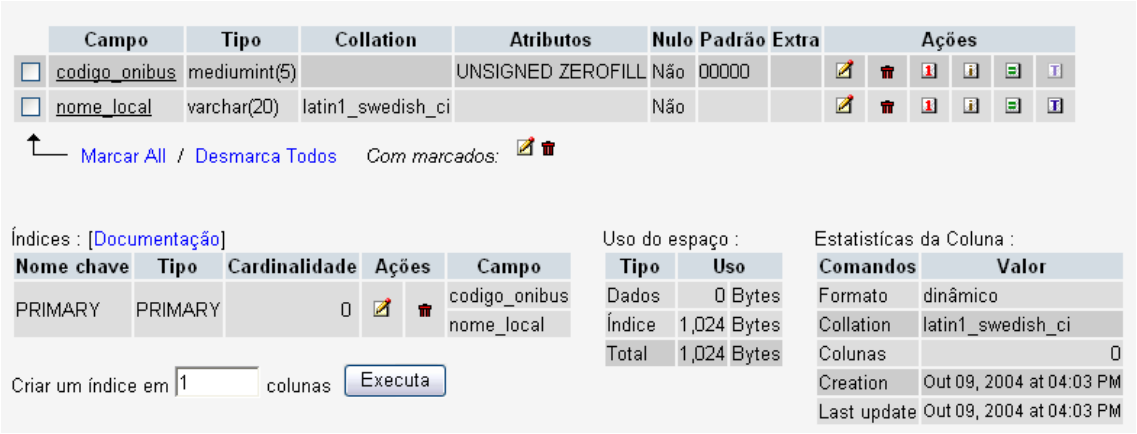


Figura 23 – Modelo final de entidades e relacionamentos do banco de dados.

No banco de dados para levantamento de requisitos não foi feita nenhuma referência a chaves estrangeiras.

Depois de povoados os caminhos intermediários para as onze linhas o programa deveria estar adaptado para se chegar em todos os locais intermediários das linhas.



Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ações
codigo_onibus	mediumint(5)		UNSIGNED ZEROFILL	Não	00000		
nome_local	varchar(20)	latin1_swedish_ci		Não			

Marcar All / Desmarca Todos Com marcados:

Índices : [Documentação]

Nome chave	Tipo	Cardinalidade	Ações	Campo
PRIMARY	PRIMARY	0		codigo_onibus
				nome_local

Criar um índice em 1 colunas Executa

Uso do espaço :

Tipo	Uso
Dados	0 Bytes
Índice	1,024 Bytes
Total	1,024 Bytes

Estatisticas da Coluna :

Comandos	Valor
Formato	dinâmico
Collation	latin1_swedish_ci
Colunas	0
Creation	Out 09, 2004 at 04:03 PM
Last update	Out 09, 2004 at 04:03 PM

Figura 24 - Tabela intermediários no banco de dados mySQL.

A tabela intermediários quando povoada ficou do seguinte modo:

←T→	codigo_onibus	nome_local
	00120	Gama D'Eça
	00121	Mauro Ramos
	00131	Agronômica
	00131	Gama D'Eça
	00132	Agronômica
	00132	Gama D'Eça
	00132	H.I.
	00133	Agronômica
	00133	Mauro Ramos
	00134	Beira-Mar Norte
	00169	Mané Vicente
	00170	João Paulo
	00170	Monte Verde
	00176	H.U.
	00176	João Paulo
	00176	Monte Verde
	00176	UFSC

Com marcados:

Figura 25 - Tabela intermediários povoada.

3.1.2.5 Modificando as pesquisas no programa de levantamento de requisitos

Agora as pesquisas deveriam poder contar com todas as localidades intermediárias das linhas e se baseariam nos tempos de chegada no ponto de chegada estimado pela linha para determinar a melhor opção.

A interface agora também selecionaria os horários de acordo com os dias da semana especificados pelo usuário e a opção quanto à adaptação para deficientes físicos.

3.2 O Banco de Dados

O banco de dados do sistema em questão contém informações referentes às linhas de ônibus as quais serão analisadas a fim de se buscar o melhor caminho entre dois pontos de referência de horários dados por uma empresa de serviço de transporte público real.

Para o exemplo resolvido neste estudo foram utilizadas onze linhas interligando duas subestações e passando por quatro pontos de referência. Foram considerados horários de dias de semana, sábados e domingos e feriados. E a possibilidade de se utilizar ônibus que possuam adaptação para deficientes físicos.

Foi considerada a possibilidade de mais de uma empresa envolvida, apesar de ter sido utilizada uma única. Algumas outras informações não diretamente envolvidas na resolução do problema também foram adicionadas ao banco de dados.

3.2.1 A Implementação do Banco de Dados do Sistema

A abordagem utilizada no modelo de banco de dados foi o modelo entidade relacionamento de banco de dados, quando da fase inicial de elaboração do projeto. A primeira parte referente às definições iniciais do banco foi um pouco mais complicada devido à necessidade de se levantar os dados relevantes à resolução do problema. Resolvidos os problemas iniciais quanto ao modelo este foi utilizado para que fosse possível se criar definições para tabelas no modelo relacional. As quais serviram de base para os documentos subseqüentes.

Foi utilizado o banco de dados desenvolvido com o protótipo.

3.3 Especificação do Preguiça

Para o sistema de busca de melhor caminho em uma rede de transporte comunitário voltado para dispositivos móveis foi escolhida a abordagem de programação extrema como uma forma de modelagem que permitisse que os resultados fossem verificados a medida em que o sistema seria desenvolvido.

Não sendo um sistema grande, e sendo desenvolvido como um trabalho de conclusão de curso, este sistema não foi desenvolvido em equipes como descrito no formato de engenharia de software da abordagem extrema. Também foi desenvolvido por apenas um programador, o que dificultou o intercambio de idéias relativas ao código gerado. Porém mesmo assim a abordagem extrema foi considerada a melhor para a modelagem do sistema devido à sua flexibilidade – o que inclui a possibilidade de modificação da forma como é abordada, ou seja, a forma flexível da abordagem possibilitou que o trabalho fosse adaptado à programação individual.

O formato do banco de dados já havia sido modelado durante algumas iterações com o protótipo do sistema utilizado para o levantamento dos requisitos iniciais – como pôde ser verificado nas sessões anteriores.

Partindo-se inicialmente dos problemas descritos pelas necessidades de usuários deste sistema de transporte comunitário definiu-se quais seriam as funcionalidades que precisariam ser implementadas em cada etapa durante o processo de modelagem e desenvolvimento de software segundo a abordagem de programação extrema.

Os conceitos necessários para o entendimento do desenvolvimento do sistema e tecnologias nele utilizadas estão explicitados em capítulos anteriores.

3.3.1 Definição dos User Stories iniciais

Segundo a abordagem extrema o primeiro passo a se seguir no desenvolvimento de um sistema se daria pelo levantamento das necessidades especificadas pelo usuário final do sistema, para então organizá-las em etapas de desenvolvimento segundo sua

importância, prioridade e em grupos coerentes que poderiam então ser desenvolvidos e lançados em versões incrementais do sistema.

Ao se verificar que as necessidades básicas dos clientes finais seriam basicamente pesquisas específicas ao dados optou-se por primeiro se definir todas as pesquisas para que se pudesse ter na primeira iteração uma interface gráfica para celular que demonstrasse – ainda que sem funcionalidades implementadas – como viária a ser o sistema final em si, para quem sabe se submeter esta interface a uma nova avaliação, o que provavelmente recairia sobre a remodelagem de alguns user stories.

Como o projeto seria desenvolvido por apenas um desenvolvedor decidiu-se por após esta primeira etapa, se necessário, remodelar-se o banco de dados e em seguida se preparar a parte de persistência de objetos relativos ao banco de dados para então partir-se para as próximas etapas de desenvolvimento iterativo.

3.3.1.1 User Stories relativos a pesquisas para definição de uma interface com o usuário

As primeiras pesquisas a serem desenvolvidas se baseariam no bairro de onde se quer sair chegando-se então a um destino específico – também determinado por um local. Seria interessantes também nesta parte já incluir a opção de baldeação para se chegar a algum ponto. Então se definiu:

- Pesquisa segundo o local de saída de acordo com os ônibus que saem do local de referência no bairro em direção ao próximo local de referência;
- Pesquisa de um ponto a outro com a utilização de baldeação de ônibus.

Assim no menu inicial já definiríamos a opção de saída segundo um ponto de referência. Pensando numa linha específica chegou-se também a mais um item:

- Pesquisas de horários segundo linhas explicitadas pelo cliente.

Afinal o cliente poderia por exemplo querer apenas saber o horário de uma linha específica. Porém o ônibus poderia estar indo ou vindo em direção ao local onde o cliente esta. Então esta pesquisa foi redefinida como:

- Pesquisas de horários segundo linhas explicitadas pelo cliente e local de saída da linha.

Então chegamos à conclusão de que as opções iniciais do menu seriam “pesquisa simples” e “pesquisa de melhor caminho e horário”. Sendo que em pesquisa simples seria explicitado o horário aproximado que se gostaria de saber se a linha tem saída de um local específico e em pesquisa de melhor caminho e horário devolver-se-ia uma seqüência de linhas necessárias a se chegar de um ponto a outro no menor tempo possível entre dois pontos de referência.

As pesquisas deveriam também ser explicitadas segundo a hora atual, ou por algum horário definido pelo usuário. Então definiu-se:

- Pesquisas com base no horário atual ou uma pesquisa em um horário definido pelo usuário.

Como dito anteriormente o sistema nada mais é do que um sistema simples de pesquisa desenvolvido para dispositivos móveis utilizando algumas tecnologias específicas para partes do sistema. Ao fim do trabalho pretende-se utilizar esta possível aplicação comercial para demonstrar uma gama de outros trabalhos que poderiam ser desenvolvidos para aplicações comerciais relativas a dispositivos móveis.

Agora estava-se apto para o primeiro ciclo que deveria testar a aceitação da interface e das pesquisas que iriam se desenvolver no banco de dados.

3.3.1.2 Verificação das necessidades do usuário segundo uma interface gráfica definida em dispositivo móvel

Com base nas pesquisas requeridas pelo usuário então fez uma interface para que ele tivesse certeza de que eram estes tipos de pesquisa que realmente deveriam ocorrer quando da utilização do sistema.

Figura 26 - Tela inicial e Tipos de pesquisa.

Foram utilizadas listas simples para implementar as funcionalidades neste primeiro teste de como seria a interface para o usuário do sistema.

Figura 27 - Tela de pesquisa de melhor caminho

Para a versão experimental do programa decidiu-se por se fazer a utilização da pesquisa baseada em horário atual apenas. O menu de pesquisa simples possui as mesmas opções do menu de opções de busca de melhor caminho.

Durante esta fase se percebeu que seria útil também a existência de uma opção de pesquisa que selecionasse não apenas a hora, mas também o dia – no caso dias de semana, sábados ou domingos, para o caso de consulta para outros dias.

Como qualquer passo a partir daqui dependeria do banco de dados não se pode avaliar a funcionalidade de futuras telas com dados – afinal todas elas teriam que acessar a listagem de locais disponíveis. Foi decidido por se implementar a funcionalidade de pesquisa simples na próxima versão.

3.3.2 – Definição dos objetos persistentes

Os objetos definidos nesta fase são os objetos que resultam de buscas no banco de dados e são criados no servidor que se comunica com o dispositivo celular.

Definiu-se uma interface para um objeto de pesquisa definido como ônibus, e a partir de uma implementação desta interface chegou-se a definição do principal objeto da camada de persistência, no qual se baseariam as pesquisas. Os objetos relativos a caminhos nada mais eram do que vetores que continham instâncias destes objetos.

As classes que resultam de pesquisas especificas foram definidas em cinco classes que tem seu nome iniciado com a palavra Pesquisa. Estas classes foram definidas para serem chamadas através dos servidores que se comunicariam com os dispositivos finais.

3.3.3 – Definição dos servidores e das threads de acesso

Devido a impossibilidade da utilização de objetos serializaveis se utilizou de servidores que pudessem enviar informações de um modo alternativo para os dispositivos.

Cada servidor era referente a um tipo de pesquisa como definido no modelo de objetos persistentes, fazendo conexão via socket com as threads de conexão executadas no celular.

Foi desenvolvida uma thread para cada caso de pesquisa porque os dados precisariam ser tratados de modo diferente para que se pudesse utilizar a informação como desejado.

3.3.4 – Pesquisa simples

A pesquisa simples precisa utilizar inicialmente a pesquisa por linhas disponíveis no servidor para que o cliente pudesse decidir por qual linha desejaria realizar a consulta. Depois de selecionada a consulta era utilizada a thread que buscava os locais de saída do ônibus especificado, por fim era escolhido se era dia de semana, sábado ou domingo e se o ônibus precisaria ser adaptado para deficientes físicos.

O sistema de busca por pesquisa simples se baseou no horário atual para a realização da pesquisa para validar o sistema.

Figura 28 - Tela de pesquisa simples.

Figura 29 - Tela de seleção de linha e seleção de local

Figura 30 - Tela de seleção de dias e adaptação.

Figura 31 - Exemplo de resultado de pesquisa simples.

As figuras acompanham uma pesquisa por busca do próximo ônibus de código 168 a sair a partir de 16:29 do local SM Florêncio em dia de semana sem que seja necessário procurar por adaptação à deficiente físico.

3.3.5 – Pesquisa por melhor caminho entre dois pontos

A pesquisa por melhor caminho utiliza a pesquisa por todos os locais de possível saída e chegada para definir o melhor caminho entre dois pontos.

É inicialmente feita a pesquisa dos locais para que o usuário defina os pontos inicial e final do caminho, depois ele define se a pesquisa se refere a dia de semana e se é preciso se buscar por um ônibus adaptado.

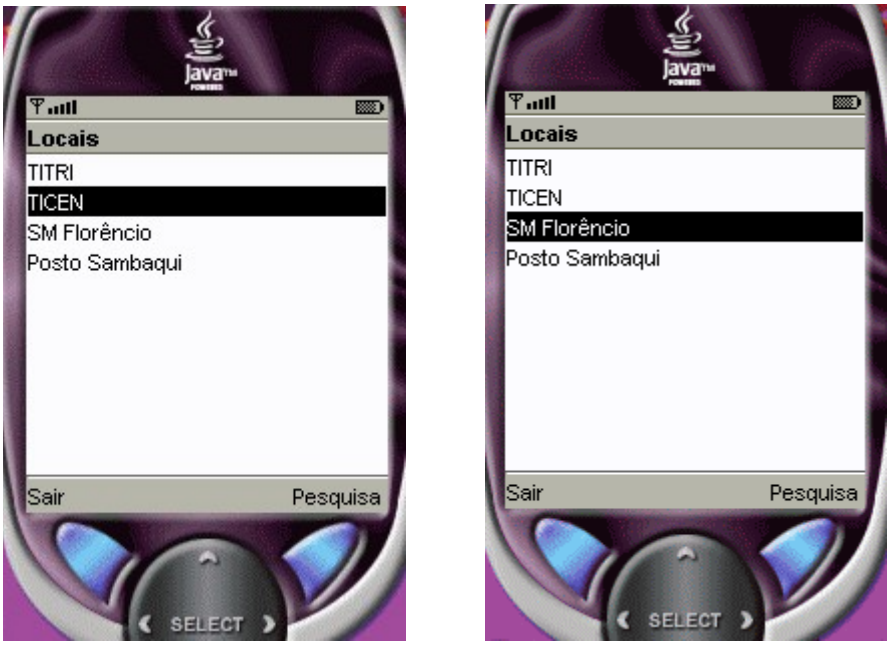


Figura 32 - Tela de seleção de local de saída e local de chegada

Como exemplo é feita uma pesquisa entre o local TICEN e o local SM Florêncio em dias de semana sem busca por adaptação a partir de 16:38.

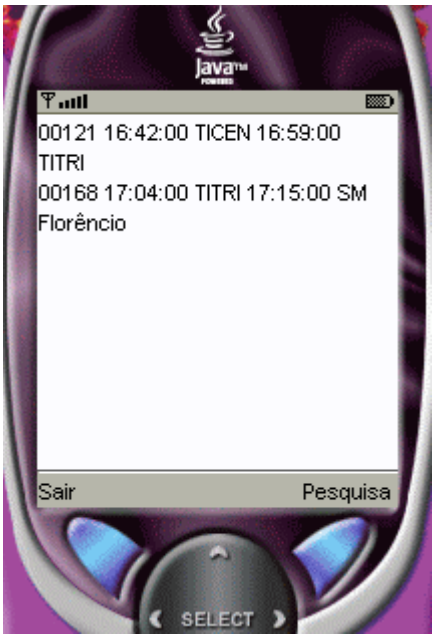


Figura 33 - Exemplo de resultado de pesquisa por melhor caminho.

O resultado mostra a conexão entre as linhas 121 e 168 passando pelo ponto intermediário TITRI.

3.3.6 – Adaptação para Ip na fase de testes

Para facilitar os testes em dispositivos para uma iteração real feita em uma rede local a tela inicial foi adaptada para que se pudesse entrar com o numero do ip da maquina que estaria servido os recursos referentes a pesquisa em banco de dados.

Figura 34 - Tela inicial modificada com entrada para ip.

3.3.7 – Interface final

Para a interface final desenvolvida para o usuário decidiu-se pela utilização de objetos desenvolvidos especificamente para aplicação de forma a tornar a interface mais próxima das interfaces utilizadas para, por exemplo, se desenvolver jogos em dispositivos celulares.

Figura 35 - Pesquisa simples interface final

Figura 36 - Pesquisa de caminho interface final

Capítulo 4 – Trabalhos Futuros e Conclusões

Este estudo teve como finalidade a modelagem de um sistema de pesquisa de melhor horário num sistema de transporte público envolvendo baldeações entre linhas e considerando locais intermediários por onde as linhas passam para encontrar a solução de melhor caminho para este problema.

O trabalho visava demonstrar a utilidade de serviços empregados em computação móvel do tipo celular, como serviços de utilidade diária. No problema apresentado demonstrou-se como poderia se ajustar informações no celular para que além de buscar pelos melhores caminhos e também para pesquisa de uma linha específica.

Para a verificação da possibilidade de desenvolvimento do sistema foi feito inicialmente um protótipo em ambiente desktop para se verificar as necessidades iniciais do usuário e para que pudesse ser verificados quais dados seriam necessários ser armazenados no banco de dados para o sistema em questão desenvolvido.

O trabalho foi desenvolvido com a utilização do ambiente para desenvolvimento de aplicações Java denominado Netbeans na versão 3.6, por ser uma ferramenta que integra várias outras ferramentas facilitando o desenvolvimento do sistema como um todo. Foi utilizada a ferramenta Java 2 Micro Edition Wireless Toolkit versão 2.1 em conjunto com o Netbeans para se desenvolver os aplicativos da parte de tecnologias móveis relativo à computação celular.

Por ser um sistema simples de pesquisa não existiu a necessidade de preocupação quanto a segurança dos dados acessados, porém para que um sistema fosse validado quanto a parte de modificação de dados por acesso por um sistema celular poderia-se utilizar um framework de comunicações seguras para dispositivos móveis que utilizasse conexões SSL, o que se recomenda para trabalhos futuros.

Como trabalhos futuros também se indica que a partir do sistema criado seja criado um framework de acesso a dados com informações de ônibus baseado no modelo

do banco de dados desenvolvido para que seja possível se criarem outras aplicações em cima das funcionalidades já existentes, como no caso da utilização de um framework relacionado à segurança para que o sistema possa também ser acessado para modificação de dados.

Alguns dados adicionais estão presentes na modelagem do sistema justamente para se justificar trabalhos na área de segurança para acesso e modificação dos dados, ou mesmo novas aplicações para usuários mais específicos do sistema – que não usuários do sistema de transportes coletivo.

O sistema também considera a possibilidade de existência de várias empresas de ônibus, e tarifas diferenciadas. Para trabalhos futuros poderia-se testar estas funcionalidades já disponibilizadas pelo sistema para procura de melhor caminho de uma forma inteligente.

Referências Bibliográficas

[CARNIEL 2003] CARNIEL, Juliano & TEIXEIRA, Clóvis. Apostila de J2ME 1.0. CEFET-PR. 2003.

[EXTREME 2004] Disponível em: http://www.extremeprogramming.org. Acesso em: 10 de outubro de 2004

[FOWLER 2003] FOWLER, Martin. Refactoring Home Page. Disponível em: http://www.refactoring.com. Acesso em: 1 de dezembro de 2003.

[CINNÉIDE 2003] Ó CINNÉIDE, M. & NIXON, P. Composite Refactorings for Java Programs. Disponível em: http://www.cs.ucd.ie/staff/meloc/home/papers/ECOOP00.pdf. Acesso em: 1 de dezembro de 2003.

[TOKUDA 2001] TOKUDA, Lance & BATORY, Don. Envolving Object-Oriented Designs with Refactorings. Disponível em: ftp://ftp.cs.utexas.edu/pub/predator/ase2000.pdf. Acesso em: 1 de dezembro de 2003.

[CASTOR 2003] CASTOR, Fernando. Refactoring. Disponível em: http://www.di.ufpe.br/~fjclf/apresentacoes/refactoring.ppt. Acesso em: 1 de dezembro de 2003.

[PRESSMAN 1995] PRESSMAN, Roger S. Engenharia de Software. São Paulo:Makron Books, 1995.

[DATE 2000] DATE, C.J. Introdução a Sistemas de Banco de Dados. Rio de Janeiro: Campus, 2000.

Anexo I – Programa em c para extração de dados

```
#include <stdio.h>
#include <stdlib.h>

void lerCaracteres(FILE *arquivo, int n, char *buf);
void anularPrimeiroCaracter(char *entrada);

int main(int argc, char *argv[])
{
   FILE *entrada, *saida;
   char buf[30]="", temp[10], hora[6], terminal[200];
   int numeroDoOnibus, diaDaSemana, adaptado = 0, i = 0;

   if (argc<3) {
      printf("%s <arquivo-de-entrada> <arquivo-de-saida>\n", argv[0]);
      return 0;
   }

   if ((entrada = fopen(argv[1], "r")) == NULL) { /* abre arquivo para leitura somente */
      perror("Erro ao abrir o arquivo de entrada: ");
      return -1;
   }

   if ((saida = fopen(argv[2], "w+")) == NULL) { /* trunca o arquivo e abre pra escrita
*/
      perror("Erro ao abrir o arquivo de saida: ");
      return -1;
   }

   printf("- Digite o numero do onibus: ");
   lerCaracteres(stdin, 9, temp);
```

```
    numeroDoOnibus = atoi(temp);

    printf("- Digite o dia da semana: ");
    lerCaracteres(stdin, 9, temp);
    diaDaSemana = atoi(temp);



    while(!feof(entrada)) {
       lerCaracteres(entrada, 6, hora);
       if (hora[0] == ' ') { anularPrimeiroCaracter(hora); }

       lerCaracteres(entrada, 200, terminal); /* ergh */
       if ((terminal[strlen(terminal)-3] == '-') && (terminal[strlen(terminal)-1] == 'D')) {
                adaptado = 1;
                terminal[strlen(terminal)-4] = 0x00;
             }
       else {
             adaptado = 0;
       }
       if (terminal[0] == ' ') {
          anularPrimeiroCaracter(terminal);
       }
          fprintf(saida, "INSERT INTO onibus(codigo_onibus) VALUES
('%d');\n",numeroDoOnibus);
       fprintf(saida, "INSERT INTO saida (codigo_onibus, nome_local, dias, horario,
adaptado) VALUES ('%d', '%s', '%d', '%.5s:00', '%d');\n",
    numeroDoOnibus, terminal, diaDaSemana, hora, adaptado);

       lerCaracteres(entrada, 6, hora);
       if (hora[0] == ' ') {
          anularPrimeiroCaracter(hora);
       }
       lerCaracteres(entrada, 200, terminal); /* ergh */
       if (terminal[strlen(terminal)-1] == '*') {
```

```
            terminal[strlen(terminal)-1] = 0x00;
        }
        if (terminal[0] == ' ') {
            anularPrimeiroCaracter(terminal);
        }

        fprintf(saida, "INSERT INTO chegada (codigo_onibus, nome_local, dias, horario,
adaptado) VALUES ('%d', '%s', '%d', '%.5s:00', '%d');\n",
    numeroDoOnibus, terminal, diaDaSemana, hora, adaptado);
        if (!feof(entrada)) {
                lerCaracteres(entrada, 1, temp); /* enjambre - pula uma linha*/
                }
    }
    puts("Finalizado!");
}

void lerCaracteres(FILE *arquivo, int n, char *buf) {
    char buffer[1024];
    int i = 0;

    fgets(buffer, 1024, arquivo);
    i = strlen(buffer);
    if (i<n) {
        n = i;
    }
    snprintf(buf, n, buffer);
    if(buf[n-1] == '\n') {
        buf[n-1] = 0x00;
    }
    buf[n] = 0x00;
}
void anularPrimeiroCaracter(char *entrada) {
    char buf[1024];
    int i, x = 0;
```

```
    for(i=1; i<strlen(entrada); i++) {
        buf[x++] = entrada[i];
    }
    buf[x] = 0x00;
    sprintf(entrada, "%s", buf);
}
```

Anexo II – org.cafeina.preguiça.ambiente.Fonte AmbienteX.java

```
/*
 * AmbienteX.java
 *
 * Created on 29 de Setembro de 2004, 11:45
 *
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 */

package org.cafeina.preguiça.ambiente;

import javax.swing.JOptionPane;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.Statement;
import java.sql.Connection;
import javax.swing.DefaultComboBoxModel;
import java.util.Vector;
import org.cafeina.preguiça.rmi.*;
import org.cafeina.preguiça.om.*;
/*import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.io.*;
import java.net.*;*/


/**
 * Interface para levantamento de necessidades de um cliente de um
sistema de transporte público
 * @author Ramon Hugo
 * @since 2004-09-29
 * @version 1.0
 */

public class AmbienteX extends javax.swing.JInternalFrame {

    /** Creates new form AmbienteX */
    public AmbienteX() {
        initComponents();

        inicializaHorario();
        inicializaLocais();


    }

    private void inicializaLocais()
    {
         try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT l.nome_local FROM saida l UNION
SELECT i.nome_local FROM intermediarios i";
            ResultSet rs = stmt.executeQuery(query);
            boolean continua = true;
            int count =0;
            Vector elementos = new Vector();
            while (rs.next() & continua) {
                String a = rs.getString(1);
                elementos.addElement(a);
            }
            jComboBox1.setModel(new
javax.swing.DefaultComboBoxModel(elementos));
            jComboBox2.setModel(new
javax.swing.DefaultComboBoxModel(elementos));

         }
         catch( Exception e)
         {
```

```
            e.printStackTrace();
         }
    }

    private void inicializaHorario()
    {
        Calendar calendario = new GregorianCalendar();
        Date hora = calendario.getTime();
        int h = hora.getHours();
        int m = hora.getMinutes();
        jComboBox3.setSelectedIndex(h-1);
        jComboBox4.setSelectedIndex(m-1);
        jComboBox5.setSelectedIndex(5);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jLabel10 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jPanel4 = new javax.swing.JPanel();
        jPanel7 = new javax.swing.JPanel();
        jPanel8 = new javax.swing.JPanel();
        jPanel12 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jComboBox1 = new javax.swing.JComboBox();
        jLabel5 = new javax.swing.JLabel();
        jComboBox3 = new javax.swing.JComboBox();
        jLabel7 = new javax.swing.JLabel();
        jComboBox4 = new javax.swing.JComboBox();
        jLabel8 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jComboBox5 = new javax.swing.JComboBox();
        jPanel13 = new javax.swing.JPanel();
        jLabel6 = new javax.swing.JLabel();
        jComboBox2 = new javax.swing.JComboBox();
        jPanel14 = new javax.swing.JPanel();
        jPanel9 = new javax.swing.JPanel();
        jRadioButton1 = new javax.swing.JRadioButton();
        jRadioButton2 = new javax.swing.JRadioButton();
        jRadioButton3 = new javax.swing.JRadioButton();
        jCheckBox1 = new javax.swing.JCheckBox();
        jPanel10 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jPanel5 = new javax.swing.JPanel();
        jPanel11 = new javax.swing.JPanel();
        jLabel11 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jPanel6 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();

        setBorder(null);
        setTitle("Interface para levantamento de necessidades");
```

```
        jPanel1.setBackground(new java.awt.Color(0, 0, 0));
        jPanel2.setBackground(new java.awt.Color(0, 0, 0));
        jPanel2.setPreferredSize(new java.awt.Dimension(700, 25));
        jLabel10.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icons/openoffice-
stock/stock_help-chat.png")));
        jPanel2.add(jLabel10);

        jLabel1.setFont(new java.awt.Font("MS Sans Serif", 0, 10));
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setText("Pregui\u00e7a System developed by Cafeina
Corp.");
        jPanel2.add(jLabel1);

        jLabel9.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icons/openoffice-
stock/stock_help-chat.png")));
        jPanel2.add(jLabel9);

        jPanel1.add(jPanel2);

        jPanel3.setBackground(new java.awt.Color(102, 102, 102));
        jPanel3.setPreferredSize(new java.awt.Dimension(50, 400));
        jPanel1.add(jPanel3);

        jPanel4.setBackground(new java.awt.Color(0, 0, 0));
        jPanel4.setPreferredSize(new java.awt.Dimension(570, 380));
        jPanel7.setBackground(new java.awt.Color(102, 102, 102));
        jPanel7.setPreferredSize(new java.awt.Dimension(550, 10));
        jPanel4.add(jPanel7);

        jPanel8.setPreferredSize(new java.awt.Dimension(530, 90));
        jPanel12.setPreferredSize(new java.awt.Dimension(500, 35));
        jLabel4.setText("Saida:");
        jPanel12.add(jLabel4);

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "TICEN", "TITRI", "SM Florêncio", "Posto Sambaqui" }));
        jPanel12.add(jComboBox1);

        jLabel5.setText("Hor\u00e1rio:");
        jPanel12.add(jLabel5);

        jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23"
}));
        jPanel12.add(jComboBox3);

        jLabel7.setText("h");
        jPanel12.add(jLabel7);

        jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21",
"22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32",
"33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43",
"44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54",
"55", "56", "57", "58", "59" }));
        jPanel12.add(jComboBox4);
```

```
        jLabel8.setText("m");
        jPanel12.add(jLabel8);

        jLabel12.setText("Varia\u00e7\u00e3o");
        jPanel12.add(jLabel12);

        jComboBox5.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30" }));
        jPanel12.add(jComboBox5);

        jPanel8.add(jPanel12);

        jPanel13.setPreferredSize(new java.awt.Dimension(500, 35));
        jLabel6.setText("Chegada:");
        jPanel13.add(jLabel6);

        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "TICEN", "TITRI", "SM Florêncio", "Posto Sambaqui" }));
        jPanel13.add(jComboBox2);

        jPanel14.setPreferredSize(new java.awt.Dimension(290, 10));
        jPanel13.add(jPanel14);

        jPanel8.add(jPanel13);

        jPanel4.add(jPanel8);

        jPanel9.setPreferredSize(new java.awt.Dimension(530, 70));
        jRadioButton1.setSelected(true);
        jRadioButton1.setText("Semana");
        jRadioButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                eSemana(evt);
            }
        });

        jPanel9.add(jRadioButton1);

        jRadioButton2.setText("S\u00e1bado");
        jRadioButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                eSabado(evt);
            }
        });

        jPanel9.add(jRadioButton2);

        jRadioButton3.setText("Domingo");
        jRadioButton3.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                eDomingo(evt);
            }
        });
```

```
        jPanel9.add(jRadioButton3);

        jCheckBox1.setText("Adaptado");
        jPanel9.add(jCheckBox1);

        jPanel4.add(jPanel9);

        jPanel10.setBackground(new java.awt.Color(0, 0, 0));
        jPanel10.setPreferredSize(new java.awt.Dimension(550, 200));
        jScrollPane1.setMaximumSize(new java.awt.Dimension(530, 180));
        jScrollPane1.setMinimumSize(new java.awt.Dimension(530, 180));
        jScrollPane1.setPreferredSize(new java.awt.Dimension(530,
160));
        jTextArea1.setBackground(new java.awt.Color(0, 0, 0));
        jTextArea1.setForeground(new java.awt.Color(255, 255, 255));
        jScrollPane1.setViewportView(jTextArea1);

        jPanel10.add(jScrollPane1);

        jPanel4.add(jPanel10);

        jPanel1.add(jPanel4);

        jPanel5.setBackground(new java.awt.Color(102, 102, 102));
        jPanel5.setPreferredSize(new java.awt.Dimension(50, 400));
        jPanel11.setBackground(new java.awt.Color(102, 102, 102));
        jPanel11.setPreferredSize(new java.awt.Dimension(30, 300));
        jPanel5.add(jPanel11);

        jLabel11.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icons/openoffice-
stock/stock_hyperlink-internet-search.png")));
        jLabel11.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                eSearch(evt);
            }
        });

        jPanel5.add(jLabel11);

        jLabel3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icons/openoffice-
stock/stock_dialog-question-32.png")));
        jLabel3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                eHelp(evt);
            }
        });

        jPanel5.add(jLabel3);

        jPanel1.add(jPanel5);

        jPanel6.setBackground(new java.awt.Color(102, 102, 102));
        jPanel6.setPreferredSize(new java.awt.Dimension(700, 25));
        jLabel2.setFont(new java.awt.Font("MS Sans Serif", 0, 10));
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setText("Desenvolvido por Ramon Hugo de Souza");
        jPanel6.add(jLabel2);

        jPanel1.add(jPanel6);
```

```
        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        setBounds(100, 50, 700, 500);
    }//GEN-END:initComponents

    private void eSearch(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_eSearch
        jTextArea1.setText("");
        jTextArea1.append("Iniciando procura");
        jTextArea1.append("\n\nEste programa é apenas uma interface
para levantamento de requisitos.\n\n");

        DefaultComboBoxModel combo =
(DefaultComboBoxModel)jComboBox1.getModel();
        String localSaida =
(String)combo.getElementAt(jComboBox1.getSelectedIndex());

        combo = (DefaultComboBoxModel)jComboBox2.getModel();
        String localChegada =
(String)combo.getElementAt(jComboBox2.getSelectedIndex());

        combo = (DefaultComboBoxModel)jComboBox3.getModel();
        String horaSaida =
(String)combo.getElementAt(jComboBox3.getSelectedIndex());

        combo = (DefaultComboBoxModel)jComboBox4.getModel();
        String minutoSaida =
(String)combo.getElementAt(jComboBox4.getSelectedIndex());

        combo = (DefaultComboBoxModel)jComboBox5.getModel();
        int vSaida =
Integer.parseInt((String)combo.getElementAt(jComboBox5.getSelectedInde
x()));

        String horarioSaida = horaSaida+":"+minutoSaida+":00";
       // String horarioSaidaPesq = horaSaida+":"+
(Integer.parseInt(minutoSaida)+vSaida)+":00";

        Vector caminho = new Vector();

        int dias = 0;
        if (jRadioButton1.isSelected())
        {
            dias =0;
        }
        else if (jRadioButton2.isSelected())
        {
            dias=1;
        }
        else if (jRadioButton3.isSelected())
        {
            dias =2;
        }

        int adaptado = 0;
        if (jCheckBox1.isSelected())
        {
            adaptado=1;
        }
        /*
```

```
        RMIOnibus onibus;
        try
        {
            onibus  = (RMIOnibus)
Naming.lookup("rmi://localhost/Recurso_Caminho");
            caminho = onibus.getCaminho(horarioSaida, localSaida,
localChegada, dias, adaptado);


jTextArea1.append("Ônibus"+"\tSaida"+"\tHorario"+"\tChegada"+"\tHorari
o\n");
            Onibus o1 = (Onibus)caminho.elementAt(0);
            jTextArea1.append(o1.toString());
            String d = o1.getChegadaLocal();
            if(d != localChegada)
            {
                Onibus o2 = (Onibus)caminho.elementAt(1);
                jTextArea1.append(o2.toString());
            }

        }
        catch (MalformedURLException murle) {
            System.out.println();
            System.out.println("MalformedURLException");
            System.out.println(murle);
        }
        catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
            System.out.println(re);
        }
        catch (NotBoundException nbe) {
            System.out.println();
            System.out.println("NotBoundException");
            System.out.println(nbe);
        }
        catch (
            java.lang.ArithmeticException
                                    ae) {
            System.out.println();
            System.out.println("java.lang.ArithmeticException");
            System.out.println(ae);
        }*/


        try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.id_onibus = c.id_onibus AND s.horario > '"+horarioSaida+"'  AND
s.nome_local = '"+localSaida+"'  AND s.dias = '"+dias+"' AND
s.adaptado ='"+adaptado+"' ORDER BY s.horario";

            ResultSet rs = stmt.executeQuery(query);
            boolean continua = true;
            int count =0;
```

```
            String e ="";
            String d ="";

jTextArea1.append("Ônibus"+"\tSaida"+"\tHorario"+"\tChegada"+"\tHorari
o\n");
            while (rs.next() & continua) {
                String a = rs.getString(1);
                String b = rs.getString(2);
                String c = rs.getString(3);
                d = rs.getString(4);
                e = rs.getString(5);

                if (++count == 1)
                    continua = false;


                jTextArea1.append(a+"\t"+b+"\t"+c+"\t"+d+"\t"+e+"\n");
            }

            if(d != localChegada)
            {
                query = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.id_onibus = c.id_onibus AND s.horario > '"+e+"' AND s.nome_local =
'"+d+"'  AND c.nome_local = '"+localChegada+"' AND s.dias= '"+dias+"'
AND s.adaptado ='"+adaptado+"' ORDER BY s.horario";
                rs = stmt.executeQuery(query);
                continua = true;
                count =0;
                while (rs.next() & continua) {
                    String a = rs.getString(1);
                    String b = rs.getString(2);
                    String c = rs.getString(3);
                    d = rs.getString(4);
                    e = rs.getString(5);

                    if (++count == 1)
                        continua = false;



jTextArea1.append(a+"\t"+b+"\t"+c+"\t"+d+"\t"+e+"\n");
                }
            }
            con.close();
        }catch(Exception e)
        {
            System.out.println(e);
        }
    }//GEN-LAST:event_eSearch

    private void eDomingo(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_eDomingo
        jRadioButton1.setSelected(false);
        jRadioButton2.setSelected(false);
        jRadioButton3.setSelected(true);
    }//GEN-LAST:event_eDomingo

    private void eSabado(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_eSabado
        jRadioButton1.setSelected(false);
```

```
        jRadioButton2.setSelected(true);
        jRadioButton3.setSelected(false);
    }//GEN-LAST:event_eSabado

    private void eSemana(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_eSemana
        jRadioButton1.setSelected(true);
        jRadioButton2.setSelected(false);
        jRadioButton3.setSelected(false);
    }//GEN-LAST:event_eSemana

    private void eHelp(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_eHelp
        JOptionPane.showMessageDialog(this, "Este é um programa para
levantamento de requisitos.\n\nOs icones utilizados neste programa
foram adquiridos
em\nhttp://www.novell.com/coolsolutions/nnlsmag/features/a_icon_librar
y_nls.html\nEste ícones estão licensiados segundo a LPGL.\n\nTodo
código fonte referente ao Preguiça está licensiado segundo a licensa
LGPL.\n\n A licensa LGPL pode ser encontrada no diretorio COPYING.",
"Help", JOptionPane.QUESTION_MESSAGE);
    }//GEN-LAST:event_eHelp


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JComboBox jComboBox2;
    private javax.swing.JComboBox jComboBox3;
    private javax.swing.JComboBox jComboBox4;
    private javax.swing.JComboBox jComboBox5;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel11;
    private javax.swing.JPanel jPanel12;
    private javax.swing.JPanel jPanel13;
    private javax.swing.JPanel jPanel14;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JScrollPane jScrollPane1;
```

```
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration//GEN-END:variables



   /* private String server(String servidor)
    {
        String sentence="";
        String modifiedSentence="";
        try
        {
            Socket clientSocket = new Socket("localhost", 6789);
            DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
            BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            sentence = servidor;
            outToServer.writeBytes(sentence+"\n");
            modifiedSentence = inFromServer.readLine();
            clientSocket.close();
        }
        catch(Exception e)
        {
            System.err.println(e);
        }

        return modifiedSentence;
    }*/
}
```

Anexo II – Artigo: Acesso remoto a horários de ônibus – Uma abordagem de acesso a dispositivos móveis utilizando J2ME e MIDP

Anexo III – org.cafeina.preguiça.ambiente.Principal.java

```
/*
 * Principal.java
 *
 * Created on 11 de Maio de 2004, 18:33
 *
 *
 * Copyright 2004 Ramon Hugo de Souza
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 */

package org.cafeina.preguiça.ambiente;
import javax.swing.*;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.awt.image.BufferedImage;

/**
```

```
 * Interface principal para uso genérico.
 * @author Ramon Hugo
 * @since 2004-05-11
 * @version 1.0
 */

public class Principal extends javax.swing.JFrame {

    /** Creates new form Principal */
    public Principal() {
        initComponents();
        AmbienteX ambiente = new AmbienteX();
        adiciona(ambiente);

    }

    public void adiciona(JInternalFrame objeto)
    {
        jDesktopPane1.add(objeto);
        objeto.setVisible(true);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jPanel1 = new javax.swing.JPanel();
        jDesktopPane1 = new javax.swing.JDesktopPane();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();

        setTitle("Visualizator 2 v 1.0");
        setIconImage(getIconImage());
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt)
{
                exitForm(evt);
            }
        });

        jPanel1.setLayout(new java.awt.BorderLayout());

        jPanel1.setBackground(new java.awt.Color(0, 0, 0));
        jDesktopPane1.setBackground(new java.awt.Color(1, 1, 1));
        jPanel1.add(jDesktopPane1, java.awt.BorderLayout.CENTER);

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        jMenu1.setText("Menu");
        jMenuItem1.setText("Sair");
        jMenu1.add(jMenuItem1);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);
```

```
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-900)/2, (screenSize.height-700)/2,
900, 700);
    }//GEN-END:initComponents

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        new Principal().show();
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JDesktopPane jDesktopPane1;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables

}
```

Anexo IV – org.cafeina.preguiça.om.InterfaceOnibus.java

```
/*
 * InterfaceOnibus.java
 *
 * Created on 24 de Outubro de 2004, 17:33
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

/**
 *
 * @author  Ramon
 */
public interface InterfaceOnibus {
```

```
    public abstract String getCodigo();
    public abstract String getSaidaLocal();
    public abstract String getSaidaHorario();
    public abstract String getChegadaLocal();
    public abstract String getChegadaHorario();
}
```

Anexo V – org.cafeina.preguiça.om.Onibus.java

```
/*
 * onibus.java
 *
 * Created on 24 de Outubro de 2004, 17:30
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;



/**
 *
 * @author  Ramon
```

```
 */
public class Onibus implements InterfaceOnibus, java.io.Serializable{

    private String codigo;
    private String saidaLocal;
    private String saidaHorario;
    private String chegadaLocal;
    private String chegadaHorario;

    /** Creates a new instance of onibus */
    public Onibus()
    {

    }

    public Onibus(String codigo, String saidaLocal, String
saidaHorario, String chegadaLocal, String chegadaHorario) {
        this.codigo = codigo;
        this.saidaLocal = saidaLocal;
        this.saidaHorario = saidaHorario;
        this.chegadaLocal = chegadaLocal;
        this.chegadaHorario = chegadaHorario;
    }

    public String getChegadaHorario() {
        return chegadaHorario;
    }

    public String getChegadaLocal() {
        return chegadaLocal;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getSaidaHorario() {
        return saidaHorario;
    }

    public String getSaidaLocal() {
        return saidaLocal;
    }

    public String toString()
    {

return(codigo+"\t"+saidaLocal+"\t"+saidaHorario+"\t"+chegadaLocal+"\t"
+chegadaHorario);
    }
}
```

Anexo VI – org.cafeina.preguiça.om.PesquisaCaminho.java

```
/*
 * PesquisaCaminho.java
 *
 * Created on 24 de Outubro de 2004, 17:54
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class PesquisaCaminho {

    private Vector caminho;

    /** Creates a new instance of PesquisaCaminho */
    public PesquisaCaminho(String horarioSaida, String localSaida,
String localChegada, int dias, int adaptado) {
        caminho = new Vector();
        try
        {
            System.out.println(localChegada);
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();

            String query3 = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.id_onibus = c.id_onibus AND s.horario > '"+horarioSaida+"'  AND
s.nome_local = '"+localSaida+"'  AND c.nome_local = '"+localChegada+"'
AND s.dias = '"+dias+"' AND s.adaptado ='"+adaptado+"' ORDER BY
s.horario";
            ResultSet rs = stmt.executeQuery(query3);
            boolean continua2 = true;
            int count =0;
            String e ="";
            String d ="";

            while (rs.next() & continua2) {
                String a = rs.getString(1);
                String b = rs.getString(2);
                String c = rs.getString(3);
                d = rs.getString(4);
                e = rs.getString(5);
                System.out.println(a+"\t"+b+"\t"+c+"\t"+d+"\t"+e);
                Onibus onibus1 = new Onibus(a, b, c, d, e);
                caminho.addElement(onibus1);

                if (++count == 1)
                    continua2 = false;

            }

            if(continua2 == true)
            {

                String query = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.id_onibus = c.id_onibus AND s.horario > '"+horarioSaida+"'  AND
s.nome_local = '"+localSaida+"'  AND s.dias = '"+dias+"' AND
s.adaptado ='"+adaptado+"' ORDER BY s.horario";
                //SELECT s.codigo_onibus, s.nome_local, s.horario,
c.nome_local, c.horario FROM saida s, chegada c WHERE s.id_onibus =
c.id_onibus AND s.horario > '"+horarioSaida+"'  AND s.nome_local =
'"+localSaida+"'  AND s.dias = '"+dias+"' AND s.adaptado
```

```
='"+adaptado+"' AND c.nome_local = '"+localChegada+"' ORDER BY
s.horario UNION
                rs = stmt.executeQuery(query);
                boolean continua = true;
                count =0;
                e ="";
                d ="";
                while (rs.next() & continua) {
                    String a = rs.getString(1);
                    String b = rs.getString(2);
                    String c = rs.getString(3);
                    d = rs.getString(4);
                    e = rs.getString(5);
                    System.out.println(a+"\t"+b+"\t"+c+"\t"+d+"\t"+e);
                    Onibus onibus1 = new Onibus(a, b, c, d, e);
                    caminho.addElement(onibus1);

                    if (++count == 1)
                        continua = false;

                }

                if(!(d.equals(localChegada)))
                {
                    query = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.id_onibus = c.id_onibus AND s.horario > '"+e+"' AND s.nome_local =
'"+d+"'  AND c.nome_local = '"+localChegada+"' AND s.dias= '"+dias+"'
AND s.adaptado ='"+adaptado+"' ORDER BY s.horario";
                    rs = stmt.executeQuery(query);
                    continua = true;
                    count =0;
                    while (rs.next() & continua) {
                        String a = rs.getString(1);
                        String b = rs.getString(2);
                        String c = rs.getString(3);
                        d = rs.getString(4);
                        e = rs.getString(5);

                        if (++count == 1)
                            continua = false;


System.out.println(a+"\t"+b+"\t"+c+"\t"+d+"\t"+e);

                        Onibus onibus2 = new Onibus(a, b, c, d, e);
                        caminho.addElement(onibus2);
                    }
                }
            }
            con.close();
        } catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public int getNumeroOnibus()
    {
        return caminho.size();
    }
```

```
    public Vector getCaminho()
    {
        return caminho;
    }
}
```

Anexo VII – org.cafeina.preguiça.om.PesquisaListaLinhas.java

```
/*
 * PesquisaListaOnibus.java
 *
 * Created on 26 de Outubro de 2004, 02:24
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class PesquisaListaLinhas {

    private Vector linhas;

    /** Creates a new instance of PesquisaListaOnibus */
    public PesquisaListaLinhas() {

        linhas = new Vector();

        try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT l.codigo_onibus FROM linha l";

            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                String a = rs.getString(1);
                linhas.addElement(a);

            }

            con.close();
        } catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public Vector getLinhas()
    {
        return linhas;
    }
}
```

Anexo VIII – org.cafeina.preguiça.om.PesquisaListaLocais.java

```
/*
 * PesquisaListaLocais.java
 *
 * Created on 26 de Outubro de 2004, 02:24
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class PesquisaListaLocais {

    private Vector locais;

    /** Creates a new instance of PesquisaListaLocais */
    public PesquisaListaLocais() {
        locais = new Vector();
        try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT l.nome_local FROM local l";

            ResultSet rs = stmt.executeQuery(query);
            boolean continua = true;
            int count =0;
            String e ="";
            String d ="";
            while (rs.next()) {
                String a = rs.getString(1);
                locais.addElement(a);

            }

            con.close();
        } catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public Vector getLocais()
    {
        return locais;
    }

}
```

Anexo IX – org.cafeina.preguiça.om.PesquisaLocalSaida.java

```
/*
 * PesquisaLocalSaida.java
 *
 * Created on 27 de Outubro de 2004, 15:41
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class PesquisaLocalSaida {

    private Vector locais;

    /** Creates a new instance of PesquisaLocalSaida */
    public PesquisaLocalSaida(String codigoOnibus) {
        locais = new Vector();
        try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT DISTINCT l.nome_local FROM saida s,
local l WHERE s.codigo_onibus = '"+codigoOnibus+"' AND s.nome_local =
l.nome_local";

            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                String a = rs.getString(1);
                locais.addElement(a);

            }

            con.close();
        } catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    public Vector getLocais()
    {
        return locais;
    }

}
```

Anexo X – org.cafeina.preguiça.om.PesquisaUmOnibus.java

```
/*
 * PesquisaUmOnibus.java
 *
 * Created on 26 de Outubro de 2004, 00:21
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.om;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Connection;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class PesquisaUmOnibus {

    Onibus onibus;

    /** Creates a new instance of PesquisaUmOnibus */
    public PesquisaUmOnibus(int codigoOnibus, String horarioSaida,
String localSaida, String diaSelect, String adaptSelect) {
        onibus = new Onibus();
        try
        {
            Class.forName("org.gjt.mm.mysql.Driver");
            String url = "jdbc:mysql://localhost:3306/ramonh";
            Connection con = DriverManager.getConnection(url,
"ramonh", "ramon687");
            Statement stmt = con.createStatement();
            String query = "SELECT s.codigo_onibus, s.nome_local,
s.horario, c.nome_local, c.horario FROM saida s, chegada c WHERE
s.codigo_onibus ='"+codigoOnibus+"' AND s.id_onibus = c.id_onibus AND
s.horario > '"+horarioSaida+"'  AND s.nome_local = '"+localSaida+"'
AND s.dias = '"+diaSelect+"' AND s.adaptado = '"+adaptSelect+"' ORDER
BY s.horario";

            ResultSet rs = stmt.executeQuery(query);
            boolean continua = true;
            int count =0;

            while (rs.next() & continua) {
                String a = rs.getString(1);
                String b = rs.getString(2);
                String c = rs.getString(3);
                String d = rs.getString(4);
                String e = rs.getString(5);
                onibus = new Onibus(a, b, c, d, e);

                if (++count == 1)
                    continua = false;

            }
            con.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public Onibus getOnibus()
    {
        return onibus;
    }

}
```

Anexo XI – org.cafeina.preguiça.server.TCPServerCaminho.java

```
/*
 * TCPServer.java
 *
 * Created on 24 de Outubro de 2004, 20:18
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.server;

import java.io.*;
import java.net.*;
import org.cafeina.preguiça.om.*;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class TCPServerCaminho {

    /** Creates a new instance of TCPServer */
    public TCPServerCaminho() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception{
        String clientSentence;
        //String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while(true) {
            //capitalizedSentence="";
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();

            String horarioSaida = "";
            String localSaida ="";
            String localChegada = "";
            int dias = 0;
            int adaptado = 0;

            int tamanho = clientSentence.length();
            //int count = 0;
            //int count2 = 0;
            int countSeq =0;
            boolean teste = false;
            String temp = "";
            for(int i=0; i < tamanho; i++)
            {
                if (i%2 != 0)
                {
                    char ts = clientSentence.charAt(i);
                    //System.out.println(ts);


                    if ((ts == ',') | (ts =='\n'))
                    {
                        //count = i;
                        teste = true;
                        if (countSeq == 0)
                        {
                            horarioSaida = temp;

                        }
                        else if (countSeq == 1)
                        {
                            localSaida = temp;
                        }
                        else if (countSeq == 2)
```

```
                        {
                            localChegada = temp;
                        }
                        else if (countSeq == 3)
                        {
                            dias = Integer.parseInt(temp);
                        }
                        else if (countSeq == 4)
                        {
                            adaptado = Integer.parseInt(temp);
                        }
                        temp="";
                        countSeq++;
                    }
                    else
                    {
                        temp += ts;
                    }
                    if(teste)
                    {
                        String tt = temp;
                        //count2 = count+1;
                        teste = false;
                    }
                }

            }
            //System.out.println("fuck:"+localSaida+"
Fuck2:"+localChegada);
            PesquisaCaminho pesquisa = new
PesquisaCaminho(horarioSaida, localSaida, localChegada, dias,
adaptado);
            clientSentence="Opção Inexistente";
            Vector caminho = pesquisa.getCaminho();
            if (caminho.size() !=0)
            {
                Onibus o = (Onibus)caminho.elementAt(0);
                clientSentence= o.getCodigo()+" "+o.getSaidaHorario()
+" "+o.getSaidaLocal()+" "+o.getChegadaHorario()+"
"+o.getChegadaLocal();
                if (caminho.size() == 2)
                {
                    Onibus o2 = (Onibus)caminho.elementAt(1);
                    clientSentence += "\n"+o2.getCodigo()+"
"+o2.getSaidaHorario()+" "+o2.getSaidaLocal()+"
"+o2.getChegadaHorario()+" "+o2.getChegadaLocal();
                }
            }

            outToClient.writeUTF(clientSentence);


        }
    }

}
```

Anexo XII – org.cafeina.preguiça.server.TCPServerListaLinhas.java

```
/*
 * TCPServer3.java
 *
 * Created on 26 de Outubro de 2004, 02:24
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.server;

import java.io.*;
import java.net.*;
import org.cafeina.preguiça.om.*;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class TCPServerListaLinhas {

    /** Creates a new instance of TCPServer3 */
    public TCPServerListaLinhas() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6589);
        while(true) {
            clientSentence = "";
            Socket connectionSocket = welcomeSocket.accept();
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            PesquisaListaLinhas listaLinhas = new
PesquisaListaLinhas();
            Vector lista = listaLinhas.getLinhas();

            for(int i=0; i< lista.size(); i++)
            {
                clientSentence = (String)lista.elementAt(i)
+","+clientSentence;
            }

            outToClient.writeUTF(clientSentence);
        }
    }

}
```

Anexo XIII – org.cafeina.preguiça.server.TCPServerListaLocais.java

```
/*
 * TCPServer4.java
 *
 * Created on 26 de Outubro de 2004, 02:24
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.server;

import java.io.*;
import java.net.*;
import org.cafeina.preguiça.om.*;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class TCPServerListaLocais {

    /** Creates a new instance of TCPServer4 */
    public TCPServerListaLocais() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6489);
        while(true) {
            clientSentence = "";
            Socket connectionSocket = welcomeSocket.accept();
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            PesquisaListaLocais listaLocais = new
PesquisaListaLocais();
            Vector lista = listaLocais.getLocais();

            for(int i=0; i< lista.size(); i++)
            {
                clientSentence = (String)lista.elementAt(i)
+","+clientSentence;
            }
            //System.out.println(clientSentence);
            outToClient.writeUTF(clientSentence);
        }
    }

}
```

Anexo XIV – org.cafeina.preguiça.server.TCPServerLocalSaida.java

```
/*
 * TCPServerLocalSaida.java
 *
 * Created on 27 de Outubro de 2004, 15:41
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.server;

import java.io.*;
import java.net.*;
import org.cafeina.preguiça.om.*;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class TCPServerLocalSaida {

    /** Creates a new instance of TCPServerLocalSaida */
    public TCPServerLocalSaida() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6389);
        while(true) {
            clientSentence = "";
            Socket connectionSocket = welcomeSocket.accept();
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            clientSentence = inFromClient.readLine();

            //System.out.println("teste"+clientSentence);
            String entrada ="";
            int tamanho = clientSentence.length();
            for(int i=0; i < tamanho; i++)
            {
                if (i%2 != 0)
                {
                    char ts = clientSentence.charAt(i);
                    entrada += ts;
                }
            }


            PesquisaLocalSaida listaLocais = new
PesquisaLocalSaida(entrada);
            clientSentence = "Problemas";
            Vector lista = listaLocais.getLocais();
            if (lista.size() !=0)
            {
                clientSentence ="";
                for(int i=0; i< lista.size(); i++)
                {
                    clientSentence = (String)lista.elementAt(i)
+","+clientSentence;
                }

            }
            System.out.println(clientSentence);
            outToClient.writeUTF(clientSentence);

        }
    }

}
```

Anexo XV – org.cafeina.preguiça.server.TCPServerUmOnibus.java

```
/*
 * TCPServer2.java
 *
 * Created on 26 de Outubro de 2004, 00:19
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package org.cafeina.preguiça.server;

import java.io.*;
import java.net.*;
import org.cafeina.preguiça.om.*;
import java.util.Vector;
```

```
/**
 *
 * @author  Ramon
 */
public class TCPServerUmOnibus {

    /** Creates a new instance of TCPServer2 */
    public TCPServerUmOnibus() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception{
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6689);
        while(true) {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();

            String horarioSaida = "";
            String localSaida ="";
            String diaSelect="";
            String adaptSelect="";
            int codigoOnibus=0;

            int tamanho = clientSentence.length();
            int countSeq =0;
            boolean teste = false;
            String temp = "";
            for(int i=0; i < tamanho; i++)
            {
                if (i%2 != 0)
                {
                    char ts = clientSentence.charAt(i);
                    if ((ts == ',') | (ts =='\n'))
                    {
                        //count = i;
                        teste = true;
                        if (countSeq == 0)
                        {
                            codigoOnibus = Integer.parseInt(temp);

                        }
                        else if (countSeq == 1)
                        {
                            horarioSaida = temp;
                        }
                        else if (countSeq == 2)
                        {
                            localSaida = temp;
                        }
                        else if (countSeq == 3)
                        {
                            diaSelect = temp;
                        }
                        else if (countSeq == 4)
```

```
                        {
                            adaptSelect = temp;
                        }
                        temp="";
                        countSeq++;
                    }
                    else
                    {
                        temp += ts;
                    }
                    if(teste)
                    {
                        String tt = temp;
                        teste = false;
                    }
                }
            }
            PesquisaUmOnibus pesquisa = new
PesquisaUmOnibus(codigoOnibus, horarioSaida, localSaida, diaSelect,
adaptSelect);
            Onibus o = pesquisa.getOnibus();
            if(!(o.equals(null)))
            {
                clientSentence= o.getCodigo()+" "+o.getSaidaHorario()
+" "+o.getSaidaLocal()+" "+o.getChegadaHorario()+"
"+o.getChegadaLocal();
            }
            outToClient.writeUTF(clientSentence);
        }
    }

}
```

Anexo XVI – ramon.Principal.java

```
/*
 * Principal.java
 *
 * Created on 24 de Agosto de 2004, 14:21
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import org.cafeina.preguiça.om.Onibus;
import ramon.conexao.*;
import java.util.Date;

/**
 *
 * @author  Ramon Hugo
 * @version
 */
public class Principal extends MIDlet implements CommandListener {

    Display tela;
    Command sair, pesquisar, telaInicial, telaTipoPesq;
    String caminho;
    Form principal, resultado, selectHor1, selectHor2;
    Canvas canvas;
    List listaInicial, listaMelhorHor, listaMelhorHor2, locais,
linhas, listaDias, listaAdaptado, listaDias2, listaAdaptado2,
listaSaida;
    String local1, local2;
    String diaSelect, adaptSelect;
    String numLinha;
    String locSaida;
    String ip;
    TextField selectIp;
    int count;

    public Principal() throws Exception{

        locais = new List("", Choice.IMPLICIT);
        linhas = new List("", Choice.IMPLICIT);
        listaSaida = new List("", Choice.IMPLICIT);

        ip = "127.0.0.1";

        sair = new Command("Sair", Command.EXIT, 0);
        pesquisar = new Command("Pesquisa", Command.ITEM, 1);
        telaInicial = new Command("Voltar", Command.ITEM, 1);
        telaTipoPesq = new Command("Voltar", Command.ITEM, 1);

        count =0;

        resultado = new Form(null, null);

        try {
            Image img = Image.createImage("/furfur2.png");

            principal = new Form(null, null);

            ImageItem imageItem = new ImageItem("",
img,ImageItem.LAYOUT_CENTER, "");

            principal.append(imageItem);
            selectIp = new TextField("Ip:", ip, 20, TextField.ANY);
            principal.append(selectIp);
        } catch(Exception e) {
            System.err.println(e);
        }

        //ItemPersonalizado i2 = new ItemPersonalizado("teste",
ImageItem.LAYOUT_CENTER);
```

```
        //principal.append(i2);

        String[] menuInicial = { "Pesquisa Simples", "Pesquisa
Caminho"};
        listaInicial = new List("Tipo de Pesquisa", Choice.IMPLICIT,
menuInicial, null);
        listaInicial.addCommand(telaInicial);
        listaInicial.setCommandListener(this);

        String[] menuMelhorHor = { "Horario Atual", "Especificar
Hora"};
        listaMelhorHor = new List("Pesquisa Caminho", Choice.IMPLICIT,
menuMelhorHor, null);
        listaMelhorHor.addCommand(telaTipoPesq);
        listaMelhorHor.setCommandListener(this);

        String[] menuMelhorHor2 = { "Horario Atual", "Especificar
Hora"};
        listaMelhorHor2 = new List("Pesquisa Simples",
Choice.IMPLICIT, menuMelhorHor2, null);
        listaMelhorHor2.addCommand(telaTipoPesq);
        listaMelhorHor2.setCommandListener(this);

        String[] dias = { "Semana", "Sabado", "Domingo" };
        listaDias = new List("Dias", Choice.IMPLICIT, dias, null);
        listaDias.addCommand(telaTipoPesq);
        listaDias.setCommandListener(this);

        String[] adaptado = { "Normal", "Adaptado" };
        listaAdaptado = new List("Adaptado", Choice.IMPLICIT,
adaptado, null);
        listaAdaptado.addCommand(telaTipoPesq);
        listaAdaptado.setCommandListener(this);

        String[] dias2 = { "Semana", "Sabado", "Domingo" };
        listaDias2 = new List("Dias", Choice.IMPLICIT, dias, null);
        listaDias2.addCommand(telaTipoPesq);
        listaDias2.setCommandListener(this);

        String[] adaptado2 = { "Normal", "Adaptado" };
        listaAdaptado2 = new List("Adaptado", Choice.IMPLICIT,
adaptado, null);
        listaAdaptado2.addCommand(telaTipoPesq);
        listaAdaptado2.setCommandListener(this);

        principal.addCommand(sair);
        principal.addCommand(pesquisar);
        principal.setCommandListener(this);


        resultado.addCommand(sair);
        resultado.addCommand(pesquisar);
        resultado.setCommandListener(this);

        selectHor2 = new Form("Pesquisa Simples");
        selectHor2.append("Sistema em desenvolvimento");
        selectHor2.addCommand(sair);
        selectHor2.addCommand(pesquisar);
        selectHor2.setCommandListener(this);

        selectHor1 = new Form("Pesquisa Caminho");
```

```
        selectHor1.append("Sistema em desenvolvimento");
        selectHor1.addCommand(sair);
        selectHor1.addCommand(pesquisar);
        selectHor1.setCommandListener(this);

    }

    public void setCaminho(String c)
    {
        resultado = new Form(null, null);
        resultado.addCommand(sair);
        resultado.addCommand(pesquisar);
        resultado.setCommandListener(this);

        caminho=c;
        resultado.append(caminho);
        tela.setCurrent(resultado);
    }

    public void setLinhas(List linhas2)
    {
        this.linhas = linhas2;
        linhas.addCommand(sair);
        linhas.addCommand(pesquisar);
        linhas.setCommandListener(this);
        tela.setCurrent(linhas);
    }

    public void setLocais(List locais2)
    {
        this.locais = locais2;
        locais.addCommand(sair);
        locais.addCommand(pesquisar);
        locais.setCommandListener(this);
        tela.setCurrent(locais);
    }

    public void setSaida(List listaSaida2)
    {
        this.listaSaida = listaSaida2;
        listaSaida.addCommand(sair);
        listaSaida.addCommand(pesquisar);
        listaSaida.setCommandListener(this);
        tela.setCurrent(listaSaida);
    }

    public void startApp() {
        tela = Display.getDisplay(this);

        tela.setCurrent(this.principal);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }


    public void commandAction(Command c, Displayable d){
        if (c == this.pesquisar) {
```

```
            ip = selectIp.getString();
            tela.setCurrent(listaInicial);
        }
        if (c == this.sair) {
            this.destroyApp(true);
            this.notifyDestroyed();
        }
         if (c == this.telaInicial) {
             tela.setCurrent(principal);
         }
        if (c == listaInicial.SELECT_COMMAND) {
            if (listaInicial.getSelectedIndex() == 0) //pesquisa
simples
            {

                if(listaInicial.equals(tela.getCurrent()))
                {
                    tela.setCurrent(listaMelhorHor2);
                /*resultado = new Form(null, null);
                resultado.addCommand(sair);
                resultado.addCommand(pesquisar);
                resultado.setCommandListener(this);*/

                //ThreadMelhorCaminho melhor = new
ThreadMelhorCaminho(this);
                //melhor.start();

                //ThreadUmOnibus onibus = new ThreadUmOnibus(this);
                //onibus.start();

              //  ThreadListaLinhas linhas = new
ThreadListaLinhas(this);
              //  linhas.start();
                //tela.setCurrent(this.locais);

                //tela.setCurrent(resultado);
                }
            }
            if (listaInicial.getSelectedIndex() == 1)
            {
                if(listaInicial.equals(tela.getCurrent()))
                {
                    tela.setCurrent(listaMelhorHor);
                }
            }
        }
        if( c == linhas.SELECT_COMMAND) {
            if(linhas.equals(tela.getCurrent()))
            {

                numLinha =
this.linhas.getString(this.linhas.getSelectedIndex());

                ThreadLocalSaida saida = new ThreadLocalSaida(this,
linhas.getString(linhas.getSelectedIndex()), ip);
                saida.start();



            }
        }
```

```
        if(c == this.listaSaida.SELECT_COMMAND)
        {
            if(listaSaida.equals(tela.getCurrent()))
            {
                //não tah pronto ainda
                locSaida =
listaSaida.getString(listaSaida.getSelectedIndex());
                tela.setCurrent(listaDias2);
                /*Date data = new Date();

                long horas = (data.getTime()/(1000*60*60)-3)%24; //gmt
-3
                long minutos = (data.getTime()/(1000*60))%60;
                String hora = horas+":"+minutos+":00";

                ThreadUmOnibus onibus = new ThreadUmOnibus(this,
numLinha, hora, locSaida, "0", "0");

                onibus.start();*/
            }
        }
        if( c== this.locais.SELECT_COMMAND) {
            if(locais.equals(tela.getCurrent()))
            {

                if (count++ == 0)
                {
                    local1 =
this.locais.getString(this.locais.getSelectedIndex());
                    tela.setCurrent(locais);
                    ThreadListaLocais local1 = new
ThreadListaLocais(this, ip);
                    local1.start();
                }
                else
                {


                    local2 =
this.locais.getString(this.locais.getSelectedIndex());
                    count =0;
                    tela.setCurrent(listaDias);

                    //ThreadMelhorCaminho melhor = new
ThreadMelhorCaminho(this, local1, local2, hora, "0", "0");
                    //melhor.start();
                }
            }

        }
        if( c == listaDias2.SELECT_COMMAND)
        {
            if(listaDias2.equals(tela.getCurrent()))
            {
                diaSelect = ""+listaDias2.getSelectedIndex();
                tela.setCurrent(listaAdaptado2);
            }
        }
        if( c == listaAdaptado2.SELECT_COMMAND)
        {
            if(listaAdaptado2.equals(tela.getCurrent()))
```

```
            {
                Date data = new Date();

                long horas = (data.getTime()/(1000*60*60)-3)%24; //gmt
-3
                long minutos = (data.getTime()/(1000*60))%60;
                String hora = horas+":"+minutos+":00";

                ThreadUmOnibus onibus = new ThreadUmOnibus(this,
numLinha, hora, locSaida, diaSelect, adaptSelect, ip);
                onibus.start();

                adaptSelect = ""+listaAdaptado2.getSelectedIndex();
            }
        }
        if( c == listaDias.SELECT_COMMAND)
        {
            if(listaDias.equals(tela.getCurrent()))
            {
                diaSelect = ""+listaDias.getSelectedIndex();
                tela.setCurrent(listaAdaptado);
            }
        }
        if( c == listaAdaptado.SELECT_COMMAND)
        {
            if(listaAdaptado.equals(tela.getCurrent()))
            {
                Date data = new Date();
                long horas = (data.getTime()/(1000*60*60)-3)%24; //gmt
-3
                long minutos = (data.getTime()/(1000*60))%60;
                String hora = horas+":"+minutos+":00";

                adaptSelect = ""+listaAdaptado.getSelectedIndex();
                ThreadMelhorCaminho melhor = new
ThreadMelhorCaminho(this, local1, local2, hora, diaSelect,
adaptSelect, ip);
                melhor.start();
            }
         }
        if( c == this.telaTipoPesq )
        {

            tela.setCurrent(listaInicial);
        }
        if ( c== listaMelhorHor.SELECT_COMMAND) //melhor caminho
        {
            if(listaMelhorHor.getSelectedIndex() == 0)
            {
                if(listaMelhorHor.equals(tela.getCurrent()))
                {
                    ThreadListaLocais local1 = new
ThreadListaLocais(this, ip);
                    local1.start();
                }
            }
            if(listaMelhorHor.getSelectedIndex() == 1) //especifica
horario
            {
                if(listaMelhorHor.equals(tela.getCurrent()))
                {
```

```
                    tela.setCurrent(selectHor1);
                }
            }
        }
        if ( c== listaMelhorHor2.SELECT_COMMAND) //Onibus horario
        {
            if(listaMelhorHor2.getSelectedIndex() == 0)
            {
                if(listaMelhorHor2.equals(tela.getCurrent()))
                {
                    ThreadListaLinhas linhas = new
ThreadListaLinhas(this, ip);
                    linhas.start();
                }
            }
            if(listaMelhorHor2.getSelectedIndex() == 1) //especifica
horario
            {
                if(listaMelhorHor2.equals(tela.getCurrent()))
                {
                    tela.setCurrent(selectHor2);
                }
            }
        }
    }


}
```

Anexo XVII – ramon.Principal2.java

```
/*
 * Principal2.java
 *
 * Created on 29 de Outubro de 2004, 10:26
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.graphics.*;
/**
 *
```

```
 * @author  Ramon
 * @version
 */
public class Principal2 extends MIDlet implements CommandListener{

    private Canvas atualCanvas;
    private Display tela;
    private int atual;

    public Principal2() throws Exception {

        atual = 0;
        tela = Display.getDisplay(this);
        atualCanvas = new PesqSimples(this);
        atualCanvas.setFullScreenMode(true);

    }

    public void startApp() {
        tela.setCurrent(atualCanvas);

    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
        System.out.println("teste");
    }

    public void setAtualCanvas( Canvas novo )
    {
        atualCanvas = novo;
        atualCanvas.setFullScreenMode(true);
        tela.setCurrent(atualCanvas);
    }
}
```

Anexo XVIII – ramon.conexao.ThreadListaLinhas.java

```
/*
 * ThreadListaOnibus.java
 *
 * Created on 26 de Outubro de 2004, 02:23
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon.conexao;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import java.util.Date;
import java.util.Vector;
import ramon.Principal;



/**
 *
 * @author  Ramon
 */
public class ThreadListaLinhas extends Thread implements Runnable{

    private Principal p;
    private String ip;

    /** Creates a new instance of ThreadListaOnibus */
    public ThreadListaLinhas(Principal p, String ip) {
        this.p =p;
        this.ip =ip;
    }

    public void run()
    {
        try
        {
            SocketConnection conexaoSocket =
(SocketConnection)Connector.open("socket://"+ip+":6589");
            DataInputStream entrada =
conexaoSocket.openDataInputStream();
            String entradaStr = entrada.readUTF();

            //tratar entrada para gerar a lista

            List linhas = new List("Linhas", Choice.IMPLICIT);

            int tamanho = entradaStr.length();
            String temp = "";

            for(int i=0; i < tamanho; i++)
            {
                char ts = entradaStr.charAt(i);

                if ((ts == ',') )
                {
                    linhas.append(temp, null);
                    temp="";
                }
                else
                {
                    temp += ts;
                }
            }
            p.setLinhas(linhas);

            conexaoSocket.close();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }
```

```
}
```

Anexo XIX – ramon.conexao.ThreadListaLocais.java

```
/*
 * ThreadListaLocais.java
 *
 * Created on 26 de Outubro de 2004, 02:23
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon.conexao;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import java.util.Date;
import java.util.Vector;
import ramon.Principal;

/**
 *
 * @author  Ramon
 */
public class ThreadListaLocais extends Thread implements Runnable {

    private Principal p;
    private String ip;

    /** Creates a new instance of ThreadListaLocais */
    public ThreadListaLocais(Principal p, String ip) {
        this.ip = ip;
        this.p = p;
    }

    public void run()
    {
        try
        {
            SocketConnection conexaoSocket =
(SocketConnection)Connector.open("socket://"+ip+":6489");

            DataInputStream entrada =
conexaoSocket.openDataInputStream();
            String entradaStr = entrada.readUTF();
            //System.out.println("that:"+entradaStr);

            List locais = new List("Locais", Choice.IMPLICIT);

            int tamanho = entradaStr.length();
            String temp = "";

            for(int i=0; i < tamanho; i++)
            {
                char ts = entradaStr.charAt(i);

                if ((ts == ',') )
                {
                    locais.append(temp, null);
                    temp="";
                }
                else
                {
                    temp += ts;
                }
            }


            p.setLocais(locais);

            conexaoSocket.close();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }
```

```
}
```

Anexo XX – ramon.conexao.ThreadLocalSaida.java

```
/*
 * ThreadLocalSaida.java
 *
 * Created on 27 de Outubro de 2004, 15:42
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon.conexao;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import java.util.Date;
import java.util.Vector;
import ramon.Principal;

/**
 *
 * @author  Ramon
 */
public class ThreadLocalSaida extends Thread implements Runnable {

    private Principal p;
    private String codigoOnibus;
    private String ip;

    /** Creates a new instance of ThreadLocalSaida */
    public ThreadLocalSaida(Principal p, String codigoOnibus, String
ip) {
        this.p = p;
        this.codigoOnibus = codigoOnibus;
        this.ip = ip;
    }

    public void run()
    {
        try
        {
            SocketConnection conexaoSocket =
(SocketConnection)Connector.open("socket://"+ip+":6389");
            DataOutputStream saida =
conexaoSocket.openDataOutputStream();

            saida.writeChars(codigoOnibus);

            saida.close();

            DataInputStream entrada =
conexaoSocket.openDataInputStream();

            String entradaStr = entrada.readUTF();

            List locais = new List("Locais", Choice.IMPLICIT);

            int tamanho = entradaStr.length();
            String temp = "";

            for(int i=0; i < tamanho; i++)
            {
                char ts = entradaStr.charAt(i);

                if ((ts == ',') )
                {
                    locais.append(temp, null);

                    temp="";
                }
                else
                {
                    temp += ts;
                }
            }
```

```
            p.setSaida(locais);

            conexaoSocket.close();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }

}
```

Anexo XXI – ramon.conexao.ThreadMelhorCaminho.java

```
/*
 * ThreadMelhorCaminho.java
 *
 * Created on 25 de Outubro de 2004, 22:54
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon.conexao;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import java.util.Date;
import java.util.Vector;
import ramon.Principal;
import java.util.Calendar;

/**
 *
 * @author  Ramon
 */
public class ThreadMelhorCaminho extends Thread {

    private String caminho;
    private String local1;
    private String local2;
    private Principal p;
    private String time;
    private String dias;
    private String adaptado;
    private String ip;

    /** Creates a new instance of ThreadMelhorCaminho */
    public ThreadMelhorCaminho(Principal p, String local1, String
local2, String time, String dias, String adaptado, String ip) {
        caminho = "";
        this.local1 = local1;
        this.local2 = local2;
        this.time = time;
        this.dias = dias;
        this.adaptado = adaptado;
        this.p = p;
        this.ip = ip;
    }

    public void run()
    {
        try
        {
            SocketConnection conexaoSocket =
(SocketConnection)Connector.open("socket://"+ip+":6789");
            DataOutputStream saida =
conexaoSocket.openDataOutputStream();


saida.writeChars(time+","+local1+","+local2+","+dias+","+adaptado+",")
;
            saida.close();

            DataInputStream entrada =
conexaoSocket.openDataInputStream();
            String entradaStr = entrada.readUTF();
            caminho = entradaStr;
            //System.out.println(entradaStr);

            p.setCaminho(caminho);
            conexaoSocket.close();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }
```

```
}
```

Anexo XXII – ramon.conexao.ThreadUmOnibus.java

```
/*
 * ThreadUmOnibus.java
 *
 * Created on 26 de Outubro de 2004, 00:40
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */

package ramon.conexao;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Image;
import java.io.*;
import javax.microedition.io.*;
```

```
import java.util.Date;
import java.util.Vector;
import ramon.Principal;

/**
 *
 * @author  Ramon
 */
public class ThreadUmOnibus extends Thread implements Runnable {

    private String onibus;
    private Principal p;
    private String numLinha;
    private String time;
    private String localSaida;
    private String diaSelect;
    private String adaptSelect;
    private String ip;

    /** Creates a new instance of ThreadUmOnibus */
    public ThreadUmOnibus(Principal p, String numLinha, String time,
String localSaida, String diaSelect, String adaptSelect, String ip) {
        onibus="";
        this.numLinha = numLinha;
        this.time = time;
        this.localSaida = localSaida;
        this.diaSelect = diaSelect;
        this.adaptSelect = adaptSelect;
        this.ip = ip;
        this.p = p;
    }

    public void run()
    {
        try
        {
            SocketConnection conexaoSocket =
(SocketConnection)Connector.open("socket://"+ip+":6689");
            DataOutputStream saida =
conexaoSocket.openDataOutputStream();
            Date data = new Date();
            int horas = (int) ((data.getTime()/1440000)-12)%24;
            int minutos = (int)(data.getTime()/60000)%60;
            String hora = horas+":"+minutos+":00";


saida.writeChars(numLinha+","+time+","+localSaida+","+diaSelect+","+ad
aptSelect+",");

            saida.close();

            DataInputStream entrada =
conexaoSocket.openDataInputStream();
            String entradaStr = entrada.readUTF();
            onibus = entradaStr;
            //System.out.println(entradaStr);

            p.setCaminho(onibus);
            conexaoSocket.close();
        }
        catch(IOException e)
```

```
            {
                e.printStackTrace();
            }
        }

    }
```

Anexo XXIII – ramon.graphics.PesqCaminho.java

```
/*
 * PesqCaminho.java
 *
 * Created on 29 de Outubro de 2004, 12:37
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.pesqCaminho.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class PesqCaminho extends Canvas{

    protected Principal2 p;

    public PesqCaminho(Principal2 p)
    {
        this.p =p;
    }

    protected void paint(Graphics graphics) {
        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);
            graphics.setColor(155,255,155);
            graphics.drawString("Tipo de Pesquisa", 0, 0, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Pesquisa Simples", 42, 130, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Pesquisa Caminho", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Sair", 150, 170, 0);
        }
        catch ( Exception e )
        {

        }
    }

    protected void keyPressed( int key)
    {
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new PesqSimples(p));
        }
        if( key == -5 | key == -6 ) {
            p.setAtualCanvas(new HoraAtual(p));
        }
         if( key == -7 )
         {
            p.destroyApp(true);
            p.notifyDestroyed();
         }
    }
}
```

Anexo XXIV – ramon.graphics.PesqSimples.java

```
/*
 * InicialCanvas.java
 *
 * Created on 29 de Outubro de 2004, 10:33
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.pesqSimples.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class PesqSimples extends Canvas{

    private Principal2 p;

    public PesqSimples(Principal2 p) {
        this.p = p;
    }

    protected void paint(Graphics graphics) {

        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);
            graphics.setColor(155,255,155);
            graphics.drawString("Tipo de Pesquisa", 0, 0, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Pesquisa Simples", 42, 130, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Pesquisa Caminho", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Sair", 150, 170, 0);
        }
        catch ( Exception e )
        {

        }
    }

    protected void keyPressed( int key)
    {
        System.out.println(""+key);
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new PesqCaminho(p));
        }
        if( key == -5 | key == -6 ) {
            p.setAtualCanvas(new HoraAtual(p));
        }
        if( key == -7 )
         {
            p.destroyApp(true);
            p.notifyDestroyed();
         }
    }


}
```

Anexo XXV – ramon.graphics.pesqCaminho.HoraAtual.java

```
/*
 * PesqSimplesHorAtual.java
 *
 * Created on 29 de Outubro de 2004, 14:16
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics.pesqCaminho;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class HoraAtual extends Canvas{
    private Principal2 p;
    public HoraAtual(Principal2 p) {
        this.p = p;
    }

    protected void paint(Graphics graphics) {
        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);
            graphics.setColor(155,255,155);
            graphics.drawString("Pesquisa Caminho", 0, 0, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Hora Atual", 57, 130, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Especificar Hora", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Voltar", 145, 170, 0);
        }
        catch ( Exception e )
        {
        }
    }
    protected void keyPressed( int key)
    {
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new HoraEspecificada(p));
        }
        if( key == -5 | key == -6 ) {
            //executa
        }
        if( key == -7 ) {
            p.setAtualCanvas(new PesqCaminho(p));
        }
    }
}
```

Anexo XXVI – ramon.graphics.pesqCaminho.HoraEspecificada.java

```
/*
 * PesqSimplesHorAtual.java
 *
 * Created on 29 de Outubro de 2004, 14:16
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics.pesqCaminho;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class HoraEspecificada extends Canvas{
    private Principal2 p;
    public HoraEspecificada(Principal2 p) {
        this.p = p;
    }
    protected void paint(Graphics graphics) {

        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);

            graphics.setColor(155,255,155);
            graphics.drawString("Pesquisa Caminho", 0, 0, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Hora Atual", 57, 130, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Especificar Hora", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Voltar", 145, 170, 0);
        }
        catch ( Exception e ){ }
    }
    protected void keyPressed( int key)
    {
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new HoraAtual(p));
        }
        if( key == -5 | key == -6 ) {
            //executa
        }
        if( key == -7 ) {
            p.setAtualCanvas(new PesqCaminho(p));
        }
    }
}
```

Anexo XXVII – ramon.graphics.pesqSimples.HoraAtual.java

```
/*
 * PesqSimplesHorAtual.java
 *
 * Created on 29 de Outubro de 2004, 14:16
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics.pesqSimples;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class HoraAtual extends Canvas{
    private Principal2 p;
    public HoraAtual(Principal2 p) {
        this.p = p;
    }

    protected void paint(Graphics graphics) {

        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);
            graphics.setColor(155,255,155);
            graphics.drawString("Pesquisa Simples", 0, 0, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Hora Atual", 57, 130, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Especificar Hora", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Voltar", 145, 170, 0);
        }
        catch ( Exception e ){}
    }

    protected void keyPressed( int key)
    {
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new HoraEspecificada(p));
        }
        if( key == -5 | key == -6 ) {
            //executa
        }
        if( key == -7 ) {
            p.setAtualCanvas(new PesqSimples(p));
        }
    }
}
```

Anexo XXVIII – ramon.graphics.pesqSimples.HoraEspecificada.java

```
/*
 * PesqSimplesHorAtual.java
 *
 * Created on 29 de Outubro de 2004, 14:16
 *
 * Copyright 2004 Ramon Hugo de Souza (ramon.hugo@gmail.com)
 *
 *
 * Este arquivo faz parte do Preguiça.
 *
 * Preguiça é um software livre; você pode redistribui-lo e/ou
 * modifica-lo dentro dos termos da Licença Pública Geral Menor GNU
como
 * publicada pela Fundação do Software Livre (FSF); na versão 2.1 da
 * Licença, ou (na sua opnião) qualquer versão.
 *
 * Este programa é distribuido na esperança que possa ser  util,
 * mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a
qualquer
 * MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a
 * Licença Pública Geral GNU para maiores detalhes.
 *
 * Você deve ter recebido uma cópia da Licença Pública Geral Menor GNU
 * junto com este programa, se não, escreva para a Fundação do
Software
 * Livre(FSF) Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
USA
 *
 *
 * This file is part of Preguiça.
 *
 * Preguiça is free software; you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as
published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 * Preguiça is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
License
 * along with Foobar; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */


package ramon.graphics.pesqSimples;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import ramon.Principal2;
import ramon.graphics.*;
```

```
/**
 *
 * @author  Ramon
 * @version
 */
public class HoraEspecificada extends Canvas{
    private Principal2 p;
    public HoraEspecificada(Principal2 p) {
        this.p = p;
    }

    protected void paint(Graphics graphics) {

        try
        {
            graphics.setColor(0,0,0);
            graphics.fillRect(0, 0, 200, 200);
            Image img = Image.createImage("/furfur3.png");
            graphics.drawImage(img, 20, 0, 0);

            graphics.setColor(155,255,155);
            graphics.drawString("Pesquisa Simples", 0, 0, 0);
            graphics.setColor(255,255,255);
            graphics.drawString("Hora Atual", 57, 130, 0);
            graphics.setColor(155,155,255);
            graphics.drawString("Especificar Hora", 40, 145, 0);
            graphics.setColor(255,255,0);
            graphics.drawString("Selecionar", 10, 170, 0);
            graphics.drawString("Voltar", 145, 170, 0);
        }
        catch ( Exception e ){}
    }
    protected void keyPressed( int key)
    {
        if( key == -1 | key == -2 ) {
            p.setAtualCanvas(new HoraAtual(p));
        }
        if( key == -5 | key == -6 ) {
            //executa
        }
        if( key == -7 ) {
            p.setAtualCanvas(new PesqSimples(p));
        }
    }
}
```